The Dissertation Committee for Christopher Lyman Jones Certifies that this is the approved version of the following dissertation:

Robust Coalition Formation in a Dynamic, Contractless Environment

Committee:

K. Suzanne Barber, Supervisor

Aristotle Arapostathis

David Kendrick

Christine Julien

Sarfraz Khurshid

Robust Coalition Formation in a Dynamic, Contractless Environment

by

Christopher Lyman Jones, B.S.; M.S.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

The University of Texas at Austin

December 2009

Dedication

Dedicated to my parents, Lyman and Jean,
who believed that I was capable of great things
despite my couch potato tendencies

And to my wife Annalisa,
who gives meaning and purpose to my achievements

“I’m sure we can pull together, sir.”

Lord Vetinari raised his eyebrows. “Oh, I do hope not, I really do hope not. Pulling together is the aim of despotism and tyranny. Free men pull in all kinds of directions. It’s the only way to make progress.”

-Terry Pratchett

Acknowledgements

A PhD is the cumulation of a long and profound course of learning, and as such it's only appropriate to name those who've helped me learn so much over the past 4 years. My advisor, Dr. Suzanne Barber, taught me a great deal about the nuts and bolts of how research actually comes together, and provided me with invaluable skills for my career. My labmates Jaesuk Ahn, Suratna Budalakoti, Dave DeAngelis, Rajiv Kadaba, Hasrat Godil, and others who have passed through the LIPS Lab over the years provided an invigorating and fascinating environment to work in. Although I'm fairly certain they've helped me, taught me and entertained me more over the years than I've contributed to their lives and work, I hope the exchanges haven't been entirely one-sided. Rick Scott and Charlotte Harris supported me and the work of the lab in a peerless fashion – I could not have completed this work in the state that it is in without them. And lastly, my committee members have done double duty by not only offering their time and expertise in reviewing my dissertation, but also acting as my teachers and guides over the years of my engineering education.

Thank you all.

October 8, 2009

Robust Coalition Formation in a Dynamic, Contractless Environment

Publication No._____

Christopher Lyman Jones, Ph. D.

The University of Texas at Austin, 2009

Supervisor: K. Suzanne Barber

This dissertation focuses on robust coalition formation between selfish agents in a dynamic environment where contracts are unenforceable. Previous research on this topic has covered each different aspect of this problem, but no research successfully addresses these factors in combination. Therefore, a novel approach is required. This dissertation accordingly has three major goals: to develop a theoretical framework that describes how selfish agents should select jobs and partners in a dynamic, contractless environment, to test a strategy based on that framework against existing heuristics in a simulated environment, and to create a learning agent capable of optimally adjusting its coalition formation strategy based on the level of dynamic change found in its environment. Experimental results demonstrate that the Expected Utility (EU) strategy based on the developed theoretical framework performs better than strategies using heuristics to select jobs and partners, and strategies which simulate a centralized "manager". Future work in this area includes altering the EU strategy from an anytime strategy to a hill-climbing one, as well as further game theoretic explorations of the interactions between different strategies.

Table of Contents

Chapter 1: Introduction

Agents are rational, autonomous pieces of software that act on and interact with environments, humans, and other agents (Wooldridge and Jennings 1995). Multi-agent systems, where agents interact with other agents, are a particularly interesting field of study which has already been utilized in domains as diverse as network routing (White and Pagurek 1998), undersea exploration (Sarne and Kraus 2005) and collaboratively playing board games (Johansson 2006).

In a multi-agent system, agents must often form teams to solve certain complex tasks, either because it is more efficient to do so, or because no one agent possesses the skills necessary to completely finish a task on its own. This process of team formation is often complicated by several factors. For example, many agents are fully independent, and act only according to their perceived best interests. In a contractless environment such as the Internet these agents cannot be coerced, but rather must be compelled into joining teams. Furthermore, agent teams frequently operate in dynamic environments, where the tasks that agents work on suddenly and unpredictably change over time. However, such changes may cause existing agent teams to fracture, as agents on the team either leave for new opportunities or leave because they have less work to do on a job, and therefore less incentive to stay on.

Agents may therefore seek to form robust teams capable of pursuing their objectives in the face of both external changes, such as changes to problem requirements, and internal changes, which may include defections and failures within the team. However, although existing research in the field individually addresses the topics of selfish agents, dynamic environments and contractless environments, no research currently addresses all three topics in combination. For example, although algorithms exist that are capable of efficiently partitioning selfish agents into teams assigned to given jobs in a static environment (Kraus, Shehory et al. 2003), such algorithms are inadequate to address the

exponential number of potential job permutations in a dynamic environment. Likewise, although mechanisms exist for allowing teams to adapt to dynamic job requirements (Scerri, Farinelli et al. 2005), such teams are composed of selfless agents, and are not applicable to selfish agents which may not accept a loss of utility as tasks change. Finally, although contracting mechanisms exist that are capable of allowing selfish agents to deal with uncertain information or dynamic environments (Sandholm and Lesser 1996), such mechanisms are clearly not applicable to domains such as the Internet where contracts cannot be enforced.

Furthermore, the concept of selfish individuals operating in dynamic, contractless environments extends beyond purely agent-based systems to hybrid human/agent systems, and even to domains based entirely in human interaction. For example, freelance consultants and small teams within large consultant firms may have a certain degree of autonomy in setting priorities on what projects they work on and who they work with. Coalitions of freelance consultants possessing different skills and resources may join to solve complex problems, but these coalition members often act as peers, having no clear lines of authority between them. Accordingly, no penalties exist to dissuade a coalition member from dropping out or penalties may be less than rewards offer by other teams. Furthermore, the work requirements of a complex project may shift over time, either because of misunderstandings, communication failures, or inherent environmental dynamics. A comprehensive theory that addresses coalition formation between selfish agents in contractless, dynamic environments could therefore significantly boost productivity and utility in business environments by giving coalition members guidance on what projects to pursue and what teams to partner with. Similar potential exists in domains such as freelancers collaborating via the Internet, and autonomous drones in a distributed computing environment.

This dissertation will therefore address the following question: how do independent agents form robust teams to solve problems in dynamic, contractless environments? More specifically, how do agents maximize their utility – defined here as reward earned by successfully completing problems (or "jobs") – in dynamic, contractless environments through job and partner selection? Research to investigate this question follows:

1. Define an agent's utility within a team based on the agent's job and partner selection decisions in the context of conditions encountered in a dynamic, contractless environment.
2. Derive a strategy that maximizes the agent's utility in a dynamic, contractless environment.
3. Test and refine the developed strategy in a large-scale agent simulation across a wide variety of settings and parameters.
4. Create a learning agent capable of adjusting its team formation strategies according to prevailing environmental conditions.
5. Adapt the simulation and strategies to better simulate a real-world consulting environment, and compare the developed strategy to existing command-and-control strategies.

This dissertation posits that, in a contractless, dynamic environment, peer groups of autonomous agents can form high-utility teams by determining how to balance used and unused skills that agents bring to a team. More specifically, the immediate incentive for an agent to join with and stay on a team, represented by the skills that an agent is currently utilizing, must be balanced against the adaptive capability that an agent brings to a team, represented by the skills an agent is currently not utilizing, but which might be utilized as team membership and/or job requirements change.

1.1 Research Questions

This research will examine the following hypothesis:

In a contractless, dynamic environment, peer groups of autonomous agents can form robust, high-utility teams that balance skill set adaptability and redundancy with individual agent incentive.

More specifically, this dissertation is based on the premise that agent teams in a dynamic environment must have robust skill sets to handle changes in job requirements, agent defections and agent failures. However, in a contractless environment, individual agents must be provided enough incentive to join and stay with a team, even if each agent's skills are not being fully utilized. The hypothesis is tested by the following research questions.

1.1.1 Research Question 1

What combination of job and team selection heuristics optimizes an agent's utility in a dynamic, contractless environment? This question will examine how various job and team selection heuristics, each representing a different intuitive approach to operating in a dynamic, contractless environment, influence agent utility. For example, is it better for agents to try to build teams that have redundant skill sets, and can therefore tolerate team member defections, or is it better for agents to try to build teams that have auxiliary skill sets, and can therefore handle the sudden addition of new job requirements? Furthermore, are different heuristics preferable in environments with different levels of dynamicism?

1.1.2 Research Question 2

How is an agent's utility in a dynamic, contractless environment defined in terms of job and partner selection? This question seeks a utility theory that will theoretically quantify the relationship between an agent's job and partner selections and the agent's utility, thereby exploring beyond the experimental heuristic comparisons examined by RQ1. More specifically, while previous heuristics focused on individual concerns such as making teams adaptive, or redundant, or balancing team robustness with agent incentives, no heuristic examined how all the concerns addressed by RQ1 heuristics jointly interact

to influence agent utility. Accordingly, the utility theory provided by answering this question will address both why agents join and stay with a team, and what capabilities agents should bring to a team in a dynamic, contractless environment.

1.1.3 Research Question 3

How do agents acting in accordance with the utility theory found in RQ2 perform in contractless, dynamic environments? This question seeks to examine how successful agents acting according to a strategy derived from the theory developed in RQ2 will be in dynamic, contractless environments. More specifically, how does such a strategy perform across a range of dynamic environments, and how does the performance of the control strategy compare to the heuristics from RQ1?

1.1.4 Research Question 4

Can a learning agent adapt its job and partner selections to improve its utility based on observed environmental dynamics? More specifically, this question will address how observed job dynamics, rates of partner defection, and other factors allow an agent to adjust its selection of jobs and partners in real time. Note that while RQ3 focuses on utilizing the theory found in RQ2 to select jobs and partners, RQ4 will be more focused on experimentally determining optimal linkages between observed changes in environment and expressed changes in agent behavior.

1.1.5 Research Question 5

Can agents acting in accordance with the utility theory found in RQ2 be usefully deployed in a real world-based consulting domain? This question examines how the strategy tested in RQ3 performs when simulation parameters are adapted to more closely mirror real-world domain factors, and how the strategy from RQ2 compares in performance to more traditional, centralized strategies.

1.2 Contributions

This work will bridge the gap between several areas of agent research by providing a theoretical description of selfish agent utility in dynamic, contractless environments. As discussed in further detail in Chapter 2, although a significant amount of research has examined the coalition teaming – some consisting of selfish agents, or encountering dynamic environments or operating in contractless environments, no research has examined all factors in combinations including selfish agents operating in dynamic and contractless environments. By providing a theory that addresses all three concepts simultaneously, new comparisons and explorations of previously distinct agent domains may be achieved. More specifically, this research provides a formal definition of the potential payoffs for selecting partners and attempting to complete multi-task jobs in a dynamic, contractless environment. This formal definition is used to derive an Expected Utility (EU) strategy that is then experimentally shown to provide superior performance to existing heuristic-based strategies in a variety of environments. The EU strategy is also compared to "ideal" strategies which do not operate under EU's constraints (i.e., dynamic job requirements and contractless interactions with other agents). In comparison to these ideal strategies, the EU strategy is generally found to earn within one order of magnitude as much credit, thereby suggesting that the EU strategy is able to deal with dynamic, contractless environments by building robust teams of agents. Finally, the EU strategy is shown to provide superior performance to a centralized "manager" strategy in an experimental domain that closely approximates a real-world consulting domain.

Furthermore, in addition to team formation between agents operating in consulting domains, this research may also be utilized in emerging open problem-solving markets, such as Yahoo! Answers. These collaborative problem-solving domains are currently hampered by the threat of unreliable partners and uncertain problem requirements and information. However, by allowing software agents or humans connected only by the Internet to select the best problems to work on and partners to work with, the resulting utility of collaborative problem-solving can be greatly improved, even in the face of partner defections and changing or inaccurate problem requirements.

Finally, this work is both complementary and supplementary to existing trust and reputation mechanisms. More specifically, the EU strategy can be used to select partners to work with when no detailed trust and reputation information is available about potential partners – the EU strategy requires only a single estimate for the average reliability of all partners. This estimate can be provided through previous experience with a given domain, a worst-case boundary on the expected reliability of potential partners, or even gut instinct on the part of a human user. Accordingly, the EU strategy is supplemental to trust and reputation strategies in that it can be used during a bootstrapping phase when detailed information on specific potential partners is still being gathered, and complementary to trust and reputation strategies in that it can be used when no such information is available at all.

Chapter 2: Background

Coalition formation and negotiation between agents has long been studied in many different contexts, from largely theoretical work rooted in game theory (Davis and Maschler 1963), to protocols describing how computer systems can subcontract tasks to each other (Smith 1980), to recent work on efficiently determining how a group of multiple agents can best be partitioned to solve multiple unique tasks (Rahwan, Ramchurn et al. 2007). This chapter will focus on describing previous work on those aspects of coalition formation relevant to the idea of robust coalition formation between agents in dynamic, contract-free environments. More particularly, section 2.1 will discuss and contrast coalition formation between both selfish and selfless agents, while section 2.2 will discuss how coalitions of agents function in dynamic or unpredictable environments. Section 2.3 will focus on enforcement mechanisms for contracts between agents, as well how agents operate in environments that have no such mechanisms. Finally, section 2.4 will present a justification for robust coalition formation between agents in dynamic, contract-free environments in the context of the previously reviewed work.

2.1 Coalition formation

A significant amount of work has been done on coalition formation between agents, both selfish and selfless, in static environments. Because the environments are static, the question of contracts does not arise – merely determining the valuation of all possible actions is usually sufficient for agents, be they selfish or selfless, to plot out a course of action that all parties will adhere to. Accordingly, all the work discussed in this section deals with static environments, unless explicitly stated otherwise.

Many of the earliest and most important explorations of coalition formation between

autonomous entities predate the digital revolution and software agents. Specifically, mathematics researchers studying game theory contributed several key ideas to the study of coalitional games – that is, games between independent, welfare maximizing entities attempting to form groups and negotiate the division of a reward (e.g. a cash payoff) generated by the group's joint effort. The complexity of coalitional games is such that even a small number of players can generate a large number of potential outcomes. Accordingly, studies into coalitional games have focused less on finding single optimal outcomes or solutions, and more on defining feasible outcomes that might conceivably be reached by rational players.

For example the concept of the Core defines one or more outcomes to a coalition game where no coalition in a given partition (that is, a given division of multiple players into one or more coalitions) can improve its outcome by changing coalition membership or division of payoffs (Myerson 1991). Other key concepts in this area include Davis and Maschler's introduction of the Kernel, which defines a set of outcomes where every member of a given coalition has an equivalent amount to gain (or lose) by leaving (Davis and Maschler 1963), and the Shapley value, which defines the payoff to any individual member of a coalition as that member's *expected* marginal contribution to the coalition's overall value (Shapley 1997).

Further research has continued in this area, often focusing on different variations or extensions to the basic coalitional game. For example, Lesser and Sandholm have examined coalitions between computationally bounded agents, wherein agents lack sufficient computational power or time to fully consider all the potential ramifications of forming a specific coalition. Interestingly, they find that even in *superadditive* environments – that is, situations where adding more members to a coalition always increases the coalition's utility – rationally bounded agents may not choose to join larger coalitions (Sandholm and Lesser 1997). Alternatively, Shehory and Krauss have examined task selection in tandem with coalition formation (Shehory and Kraus 1998) and coalition formation mechanisms between agents in non-superadditive environments (Shehory and Kraus 1999).

More recently, significant work has taken place in developing algorithms to

efficiently search through the space of potential partitions for optimal coalitional values. Sandholm et al. have developed a mechanism to conduct a bounded search through the space of coalition structures (i.e. partitions) such that by searching through a given subset of the set of possible partitions, a maximum bound can be placed on the value of the best possible partition (Sandholm, Larson et al. 1999). Additional work by several authors (Rahwan, Ramchurn et al. 2007; Su, Hu et al. 2007) has extended this research to a point where *bounded*, *anytime* search through the partition space is possible. Specifically, it is now possible to bound the desired optimality of results (i.e. the degree to which the returned result approaches the optimal result) and accordingly search a decreased amount of search space, or to conduct search so that it can be stopped at any time, with the quality of the returned result increasing monotonically the longer the search is allowed to execute (Rahwan, Ramchurn et al. 2007). Still further research has been conducted in the area of coalition structure exploration utilizing genetic algorithms (Sen and Dutta 2000) or hierarchical decomposition search of the coalition space (Abdallah and Lesser 2004). It is noted that these last two pieces of work, in contrast to those before and after, utilizes selfless agents who seek primarily to increase their team's utility, as opposed to selfish agents, who seek primarily to increase their own utility.

Turning towards more technical aspects of coalition formation between software agents, we find Smith's seminal work on the Contract Net protocol, which provides a mechanism for agents to sub-contract tasks between each other, thereby allowing for the efficient division of work between two or more agents (Smith 1980). Additional work in contracting between agents has further extended this basic concept of contracting between agents. For example, Sandholm's work on OSCM contracts provides a mechanism wherein agents can subcontract tasks out singularly or in clusters, swap tasks between each other and engage in complex transactions concerning three or more agents (Sandholm 1998).

However, while such work provides the technical capability for software agents to coordinate on working on complex tasks, they do not provide guidance as to why such mechanisms would be preferable to top-down control, or address the question of what subcontracting it would be advantageous for an agent to engage in. Towards these ends,

Wellman and Wurman notably provided a justification for the utility of market-based behavior in providing an effective allocation of resources in a distributed manner (Wellman 1997), while Kraus, Shehory, and others have produced numerous papers on auction mechanisms wherein complex tasks are bid on by various agent coalitions (Kraus, Shehory et al. 2003; Kraus, Shehory et al. 2004; Manisterski, David et al. 2006). These auction mechanisms are capable of handling a wide variety of environments, including limited information, bounded rationality, and both selfish and unselfish agents.

Alternatively, other work that examines individual agent decisions in coalition formation examines how agents behave when connected to other agents via social networks. More particularly, Skyrms and Permantle provided research showing that social networks invariably arise between agents equipped with reinforcement learning when playing certain games (Skyrms and Pemantle 2004). Following along these lines, Gaston and others have done interesting work on how agents adapt social structures and form coalitions within their existing social structures (Gaston 2005) and how different types of social network structures influence coalition formation (Gaston, Simmons et al. 2004). Alternatively, Weerdt et al. have developed a mechanism for how agents in a social network can efficiently distribute incoming tasks to their neighbors (de Weerdt, Zhang et al. 2007). Work has also been done on overlapping coalitions, where agents with rare or valuable skills can be part of more than one coalition at once (Wilson and DesJardins 2007), or where multi-skilled agents may simultaneously provide different skills to different coalitions (Lin and Hu 2007).

Still other work in coalition formation has focused on very large scale environments, where hundreds, thousands, or even greater numbers of agents exist in a decentralized environment. Lerman and others have allowed agents in a large scale environment to balance exploration and exploitation of existing coalitions via a physics-based coalition formation mechanism (Lerman and Shehory 2000; Lerman and Galstyan 2003), while Tosic and Agha have presented an algorithm for bottom-up team generation based on graph structure and cliques of agents (Tosic and Agha 2005), and Sander et al. have described a mechanism for agents to efficiently distribute themselves around existing tasks in a two-dimensional environment (Sander, Peleschuk et al. 2002). Additionally,

White and Pagurek have covered the utility of ant-like "swarm" behavior of thousands of agents in adapting to dynamic optimization problems (White and Pagurek 1998). It is, of course, noted that work in all these fields is ongoing, since no one algorithm is likely to produce the perfect answer for any and all situations.

2.2 Coalitions in dynamic environments

Although coalition formation between agents in a fixed, static environment is difficult enough, a significant amount of research has focused on the formation and operation of coalitions in dynamic environments, where the coalitions must either change their actions or their memberships in response to external stimuli. The wide breadth of such research is not surprising when we consider how broad the concept of a "dynamic environment" actually is: the introduction of new tasks, the modification of existing tasks, changing relationships between agents, environmental uncertainty, bounded rationality or any combination thereof may require entirely new forms of coalition formation and operating, some of which we examine below. It is noted that all the work in this section deals with dynamic environments, either with selfish agents acting under enforceable contracts, or selfless agents acting without contracts.

One of the most prominent and easy-to-understand examples of a dynamic environment is the RoboCup rescue simulator, which simulates an ongoing, city-wide disaster and challenges cooperative teams of selfless agents to respond as effectively as possible (Kitano, Tadokoro et al. 1999). More specifically, the RoboCup Rescue domain is loosely based on the 1995 Kobe, Japan earthquake, and features multiple cooperative software agents assigned into various emergency response roles such as firefighters, police, and EMTs. These agents must rescue trapped civilians, minimize the amount of damage done by spreading fires, and clear blocked roads to allow better access to fires and civilians, all in a coordinated manner. The environment is dynamic in multiple ways: fires spread if not promptly extinguished, new fires pop up, and information is not perfect, meaning that agents' view of the situation, and therefore their response to it, changes with time. Programming teams from various academic institutions compete

against each other to develop the most effective teams, and have developed a very wide variety of solution implementations, from centrally coordinated systems which rely on prioritized rules to select tasks (Nazemi, Faradad et al. 2005) to centralized auctions and decentralized, low overhead communication mechanisms (Nair, Ito et al. 2001), to novel coordination mechanisms adapted from the RoboCup soccer domain (Lau, Reis et al. 2005).

At least some of the solutions found through the RoboCup Rescue domain have inspired other, more general mechanisms for adapting to dynamic circumstances. For example, Tambe et al. have produced several papers focusing on the STEAM framework, which allows for a team of agents to reassign roles based on environmental events (Tambe, Pynadath et al. 2000). For example, in one domain, cooperative software agents are assigned to simulated helicopters, and as a team, are assigned to transport valuable cargo over dangerous territory. Helicopters can be utilized either to transport cargo or to scout the territory, ensuring that no hazards are ahead, and removing hazards as they are perceived. STEAM manages how agents move back and forth between the scout and transport roles over the course of the mission as the situation on the ground changes (Nair, Tambe et al. 2003). STEAM has also been further adapted to focus on team persistence, so that changes to agent roles within a team can be considered in light of how reallocation may effect the team's long-term viability (Tambe and Zhang 1998).

Researchers have also done work on dynamically adapting the control of a coalition of agents in response to the current environment and changes to that environment. For example, Corkill and Lesser described a mechanism wherein cooperative agents in a distributed network determined how to best balance immediate domain-specific operating with handling broader network issues, such as routing messages between agents (Corkill and Lesser 1983). Alternatively, Martin demonstrated an Adaptive Decision Making Framework which allowed a multi-agent system to decide, based on varying criteria, whether agents would be better off working in a command-and-control structure, in a fully-cooperative peer-to-peer fashion, or completely independently from one another (Martin 2001).

As previously mentioned during the earlier discussion of RoboCup Rescue, even

when an environment is not specifically dynamic, uncertain or newly available information may still cause agents to adapt their current course of action, or specifically act with the understanding that the information being received is not complete or fully accurate. Accordingly, past research into software agents has focused on dealing with coalition formation and negotiation under uncertainly. For example, Hanna and Mouaddib have described mechanisms which allow selfish agents in an auction setting to rationally bid on tasks in an auction setting, despite only having probabilistic information on what the resource cost of completing those tasks will be (Hanna and Mouaddib 2002). Alternatively, Chalkiadadis and Boutilier have extended the concept of the Core, described above, by combining it with Bayesian learning for situations where selfish agents are not entirely certain about the capabilities of other agents. Under their mechanism, agents may join or leave coalitions based not only on expected rewards, but based on how exploring new coalitions can help improve an agent's information about potential coalition members (Chalkiadakis and Boutilier 2004). Similarly, Yokoo et al have proposed a framework for coalition formation between agents in an open, anonymous environment, wherein selfish agents have the capacity to project false identities, yet still must form working coalitions (Yokoo, Conitzer et al. 2005).

Other notable work involving agents with uncertain information has been produced by Ketchpel, which examined the difficulty and complexity of negotiations between selfish agents with different conceptions of a task or item's value (Ketchpel 1994), and the work of Kraus et al. on negotiation between agents with limited negotiation timeframes (Kraus, Wilkenfeld et al. 1995). Scully et al have produced work on coalition value calculation in environments where the metrics of value may change over time (Scully, Madden et al. 2004). Finally, Soh and Tsatsoulis have presented the concept of "hastily formed" coalitions, which allow coalitions of selfish agents to quickly form to solve a task's minimum requirements in environments where bounded rationality or uncertain information prevent in-depth determination of optimal coalitions (Soh and Tsatsoulis 2002).

2.3 Coalition enforcement mechanisms

Beyond the issues of coalition formation in dynamic environments, there are issues of enforcing coalition behavior in such environments. More specifically, it is fairly trivial to manage coalitions formed by perfectly cooperative agents – determining how agents should optimally act in such coalitions may be perfect, but cooperative agents will not try to cheat or break away from an assembled coalition. However, semi- or non-cooperative agents in any environment that is not perfectly static, with perfect information and total rationality, always have the potential to cheat, break from coalitions, or otherwise take actions contrary to their agreed-upon roles. Consequently, enforcement mechanisms for coalitions are also a fruitful area of research. Accordingly, this section discusses selfish agents acting in dynamic environments, unless explicitly stated otherwise.

One of the simplest ways to deal with exceptions occurring in agent behavior is to simply craft agreements between selfish agents that take such exceptions into consideration. For example, Raiffa broadly discusses the possibilities of contingency contracts, wherein payouts between agents are altered, or occur only when certain preconditions are met (Raiffa 1982). Faratin and Klein further delve into the potential of contingency contracts by offering a taxonomy of exceptions and describing a mechanism for dealing with exceptions, by taking the potential of service failure into account during subcontracting negotiations between agents (Faratin and Klein 2001).

However, contingency contracts face certain unavoidable limitations in practice. In all but the simplest domains, the list of examined contingencies cannot possibly be fully examined, meaning that the number of exceptions that such contracts can realistically deal with must be relatively small. Furthermore, the specific exceptions considered in contingency contracting are unavoidably domain-specific, meaning that few, if any, domain-independent theoretical principles can be drawn from studying them.

Alternatively, the concept of leveled commitment contracts avoids many of these pitfalls. Sandholm and Lesser were the first to suggest a leveled commitment protocol between agents, wherein selfish agents had the option to decommit, or opt out of an existing contract, simply by paying a predefined penalty. They further proved that such an arrangement could frequently lead to superior outcomes in comparison to absolute

contracts without the ability to exit (Sandholm and Lesser 1996). Note that leveled commitment contracts differ from contingency contracts in that these contracts are not designed with specific contingencies in mind. Rather, agents are free to decommit at any time upon determining that it is advantageous for them to do so. Sandholm has continued to do extensive research on leveled commitment contracts, examining the effect of contract prices and decommit prices (Andersson and Sandholm 2001), proving the existence of Nash equilibria for leveled contracts and providing algorithms to locate those equilibria (Sandholm, Sikka et al. 1999), and examining the effect of various protocols (e.g. simultaneous versus sequential decommitment) on the utility of leveled commitment contracts (Sandholm and Zhou 2002).

Of course, both contingency and leveled commitment contracts implicitly rest on the assumption that contracts between agents are enforceable, or that agents have reason to follow contracts even when carrying them out would lead to short-term loss of utility. Lesser and Sandholm examined unenforced contracts and broadly determined that unenforced contracts between agents are only possible when selfish agents' long-term utility for following a contract is greater than their short-term utility for breaking it. In a single discrete exchange of goods and/or services, this can only occur when agents have already accrued all related expenses prior to the actual exchange, but in a continuous exchange (e.g. a stream of information for a stream of money), or in a series of discrete exchanges, trades can be structured so that each party in the exchange has an incentive to stay within the contract (Sandholm and Lesser 1995). These results are confirmed by other researchers, such as Saha et al, who show that associating with other agents based on past history and known expertise can lead to increased utility and stability (Saha, Sen et al. 2003), and Rathod and DesJardins who find superior utility for agents in stable teams versus agents in "on-demand" teams, which assemble and disassemble as needed (Rathod and DesJardins 2004). The long term stability of agent coalitions and communities is also of key interest to the agent trust community, a broad overview of which is provided by Fullam (Fullam 2007).

In addition to contracting mechanisms, researchers have also studied the use of fault-tolerant mechanisms in agent coalitions. For example, beyond the "timeout" mechanism

originally proposed by Smith in the Contract Net framework (Smith 1980), Klein and Dellarocas proposed a set of domain-independent enforcement agents dedicated to monitoring the progress and compliance of selfish agents performing domain-specific tasks (Dellarocas and Klein 2000). Patel et al proposed a similar mechanism of agents in a virtual organization coordinating computational grid-based activities, carrying out roles such as "policemen", quality assurance agents, and reputation brokers (Patel, Teacy et al. 2005). Guessoum et al also described a system for replicating agents as needed for fault-tolerance in massive-scaled systems, based on criteria such as service criticality, number of existing agents and resource availability (Guessoum, Briot et al. 2005).

2.4 Robust coalition formation in dynamic, contract-free environments

Surprisingly, despite the large amount of research done on coalition formation between software agents, arguably no research has touched on the exact mix of robust coalition formation between selfish agents in a dynamic, contract-free environment. For example, Scerri et al have provided work on "Extreme Teams", which are large-scale teams of cooperative agents tasked to work on a large, dynamic set of tasks (Scerri, Farinelli et al. 2005). However, the protocol proposed for Extreme Team coordination is a low-communication overhead distributed constraint optimization protocol, and as such does not address issues such as non-cooperative agents, agent motivation, or reconfiguring task assignments to deal with dropped or newly added subtasks.

Similarly, Klusch and Gerber have introduced a simulation-based scheme for evaluating and developing potential coalitions of selfish agents in a dynamic environment (Klusch and Gerber 2002). Such work only offers an opportunity for coalition membership to be modified during the negotiation and calculation valuation process; contracts between agents are binding once entered into, and existing coalitions have no mechanism to adapt their behavior according to external or internal changes. Lin et al also offer a restructuring algorithm for coalition formation capable of dealing with the arrival of new agents during the team formation process (Lin, Hu et al. 2007), but again, coalition agreements are binding and no provision is offered for how teams should deal

with change once coalitions are formed.

Accordingly this gap in the research suggests a need for robust coalition formation between selfish agents in dynamic, contract-free environments. However, it should be noted that the work presented below owes a great deal to several of the broad concepts described above, particularly Tambe's work on adaptive teams, Sandholm's work on contracting issues, and many of the broad approaches used to deal with bounded rationally and uncertain information provided by several researchers in the field of agent coalition formation.

Chapter 3: Research Question 1

To best understand the domain of selfish agents in dynamic, contractless environments, it is helpful to examine the problem from many different angles. Towards that end, section 3.1 of this chapter features a qualitative examination of a simple theoretical model which illustrates many issues related to the domain, such as the computational complexity of dynamic environments and the balance between team size and team adaptability. Section 3.2 approaches the domain through a more quantitative perspective by modeling the problem environment through mathematical descriptions of agents, jobs, tasks and skills. Section 3.3 describes an agent simulation based on the models from section 3.2, applicable to multiple research questions. Section 3.4 describes the approach to research question 1, and section 3.5 provides the results of simulated experiments comparing the performance of various heuristic-based strategies.

3.1 Qualitative Approach

We begin by considering a set of agents and a set of jobs, illustrated in Figure 1 below. Agents are selfish, concerned only with maximizing their own rewards. Jobs are composed of multiple independent subtasks, each of which must be completed before the job is completed. These subtasks all fall into a set of different types, such that each different type of subtask requires a specific agent skill to be completed. Accordingly, each agent possesses skills that can be used to work on a specific type of subtask. For example, as shown in Figure 1, the first and second agents all possess a skill of type 1, which can be used to work on type 1 subtasks in the second and fourth jobs.

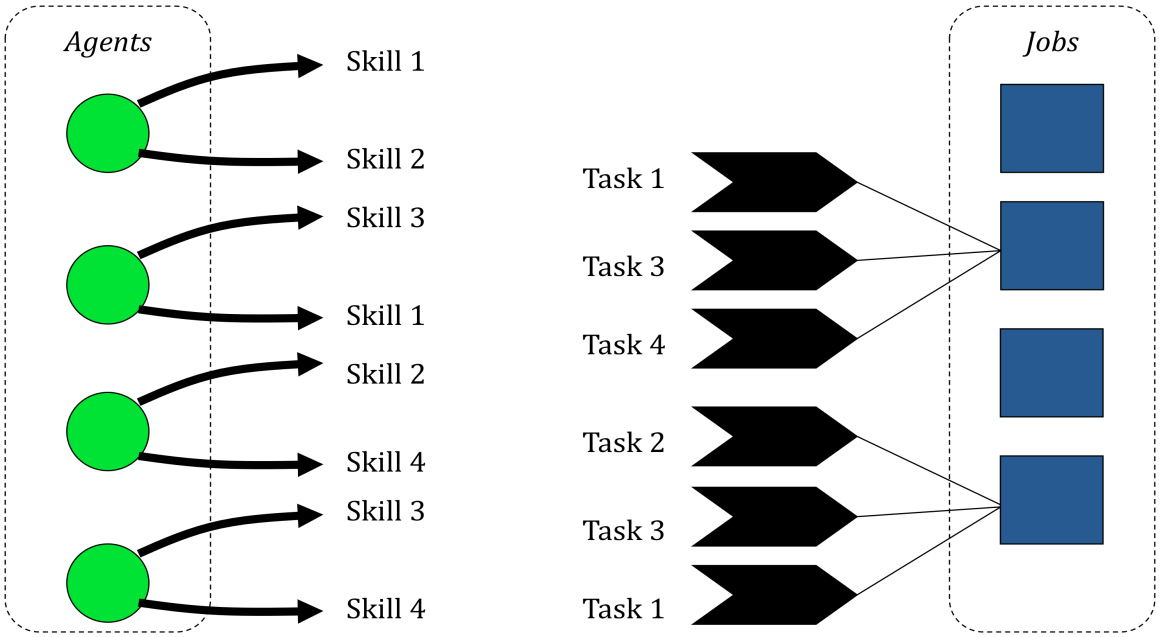


Figure 1: Agents, skills, jobs and tasks

Jobs are dynamic; meaning that as time moves forward job requirements may change. More specifically, Figure 2 shows an initial configuration of a given job, where two out of a possible six subtasks need to be completed in order for the job to be completed. However, in a dynamic environment, new subtasks may be added to the list of job requirements, and required subtasks may be dropped from the list of requirements. Accordingly, Figure 2 also shows two potential mutations of the job, the first where two additional subtasks have been added to the list of requirements, and the second where one subtask has been added and another has been dropped.

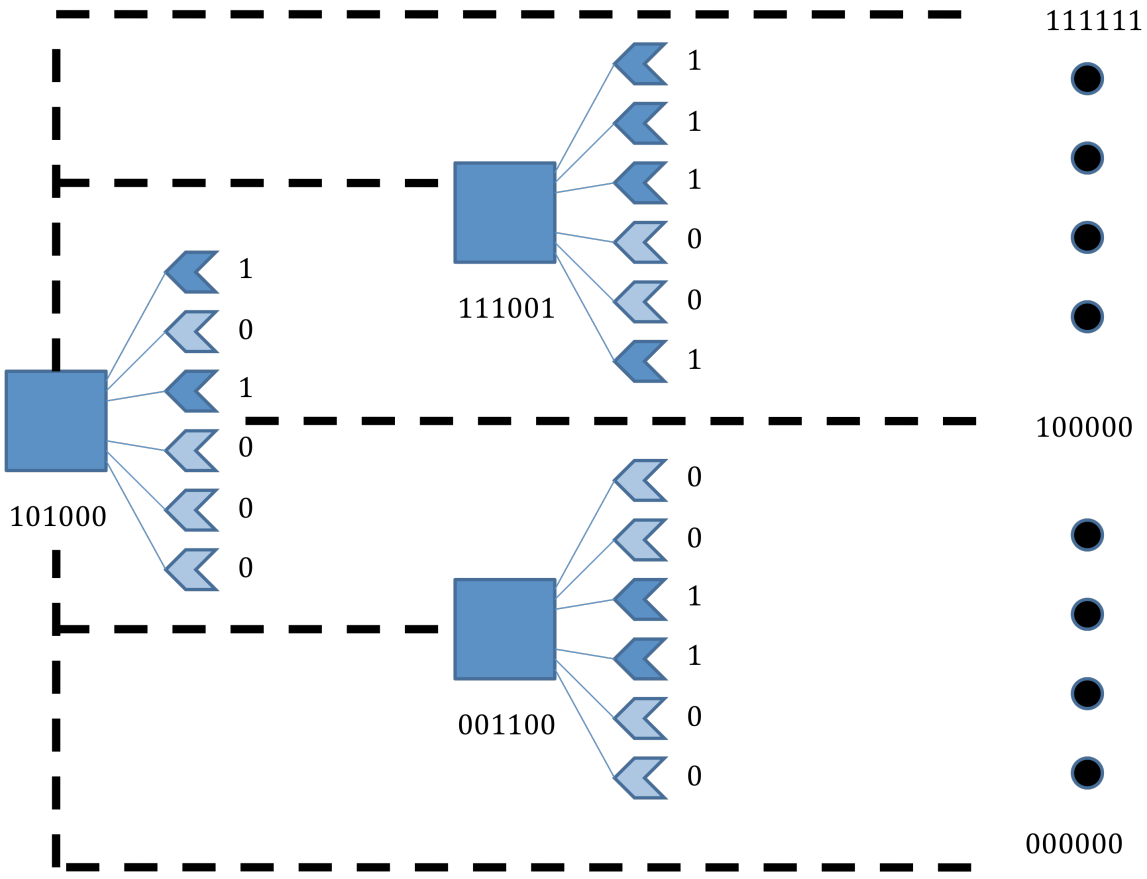


Figure 2: Job subtask dynamics

Note also that the number of permutations that a job can undergo scales exponentially with the number of jobs and potential subtasks per job. To briefly sketch this proof, note that we can assign each potential subtask in Figure 2 a one or zero depending on if the subtask is currently required for job completion. Each potential state of the job can then be described as a 2^N binary number, where N is the number of potential subtasks associated with the job. Thus, a single job with six subtasks has 64 possible permutations, and ten such jobs would have $2^{(10x6)}$ total permutations, or roughly one quintillion (10^{18}) possible states.

Furthermore, if we allow for subtasks to have more than an active/inactive state – e.g., by demanding different quality-of-service for the subtask – then a single job has exponentially more permutations. For example, if each task can demand one of three different quality of service levels in addition to potentially being inactive, the number of

permutations for a job with six subtasks is 4^6, or 4096 possible states, and ten such jobs would be $4^{(10x6)}$, or roughly 10^{36} possible states.

In such an environment, a team that wishes to complete a job even as the job requirements change must clearly possess some potential for adaptation – just because a skill is not currently required to complete a job, does not mean it will not be required in the future. However, let us make the reasonable assumption that agents earn profit in rough proportion to the direct utilization of their skills when completing a job – that is, an agent who is able to use two of its skills to complete two subtasks in a job will earn more profit than an agent who only completes one subtask. Therefore, in a dynamic, contractless environment, it is also possible that agents that are currently part of a team may leave their team if some or all of their skills are no longer required, or if another job becomes available that makes better use of an agent's available skills.

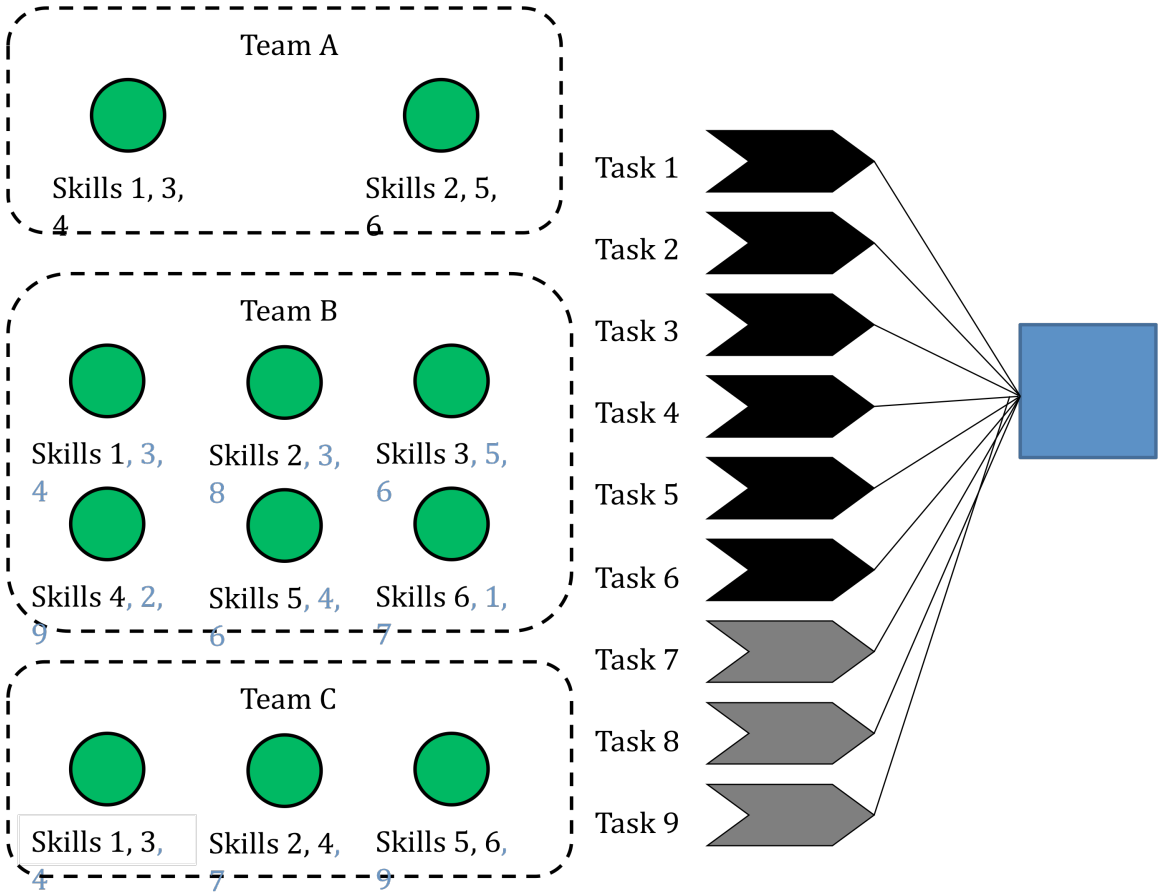


Figure 3: Potential teams and task assignments

For example, consider the three potential teams shown in Figure 3. The skills collected in Team A are currently sufficient to complete the job shown on the right, since the job only requires subtasks 1 through 6 to be completed. However, if it becomes necessary to complete any of subtasks 7, 8, or 9, then Team A will not be able to complete the job, and may potentially be unable to earn any profit.

Alternatively, Team B shows a team where only a small portion of the team's overall skill set is committed to the job on the right – those skills listed in black. The skills listed in blue are skills that might be utilized in the event that new subtasks are required, or an existing agent is no longer able to complete its assigned subtask. For example, if the first agent working on task 1 was to fail or quit, the last agent (currently working on task 6) could take over (Tambe and Zhang 1998).

However, also note that the opportunity for an agent to defect is greater with such an arrangement. Again, if we assume that agents earn profit in rough proportion to how many of their skills they are able to use, any of the six agents in Team B may find a better job to work on, where more than one of their available skills is applicable to the job. Therefore, even though Team B has a skill set capable of adapting to sudden changes, the team members have relatively little incentive to stay with the team. Were Team B to face sudden defections by multiple team members, it too could find itself lacking sufficient resources to complete the associated job.

Finally, Team C represents a team that is more optimally balanced between adaptive capacity and agent incentive. As a whole, the team possesses sufficient capacity to handle the addition of subtasks 7, 8, or 9 to the job requirements list, and each member of the team is able to use two of its available skills to work on the job. As described by earlier research, while each agent might potentially be able to increase its profit by finding another job where it was able to use all three of its available skills, such a team would find itself in a position similar to Team A, unable to adapt to new job requirements (Jones and Barber 2008).

3.2 Quantitative Problem Definition

We start our quantitative problem domain model with a set of general tasks $T = \{T_i\}$, where $1 \leq i \leq \alpha$. Each general task T_i represents a type of job that an agent might carry out: if T is limited to tasks involved in building construction, for example, T_1 might be building a driveway, while T_2 might be constructing a roof, and so on. Each general task T_i is therefore a set of task instances $\{T_{ij}\}$, where each T_{ij} is a specific instance of general task T_i associated with a job J_j, and where each job J_j is part of a set of jobs $J = \{J_j\}$, where $1 \leq j \leq \beta$. For example, if T represents the set of all tasks associated with building a building, and if J is the set of all buildings under construction, then T_{11} might be building a driveway at a first building under construction, T_{12} might be building a driveway at a second building under construction, T_{21} is constructing a roof at the first building, and so on. Alternatively, as will be further described in Chapters 4 and 5, this model is also applicable to a knowledge-based consulting domain, where different tasks are different queries requiring skills in specialized domains.

Each job J_j in set J contains a potential task instance T_{ij} of every possible task T_i in T, but only a subset of these tasks needs to be completed to finish the job. Again, returning to the building example, every building in existence could conceivably have a swimming pool, or a loading dock, or a conference room, but in practice factories and offices rarely have swimming pools, and houses rarely have loading docks. Accordingly, within each job J_j, task instances T_{ij} are separated into a set of active task instances $ActiveTasks_j$, all of which must be completed for the job to be finished, and a set of inactive task instances $InactiveTasks_j$, which are irrelevant to the job's completion status. For any job J_j, $ActiveTasks_j \cup InactiveTasks_j = \{T_{ij}\}$ for all i, and $ActiveTasks_j \cap InactiveTasks_j = \varnothing$.

Continuing on, a set of skills $S = \{S_i\}$ and a set of self-interested agents $A = \{A_k\}$ are introduced, where once again $1 \leq i \leq \alpha$ and $1 \leq k \leq \chi$. Each skill S_i is associated with a general task T_i, and may be used to work on and eventually complete any task instance T_{ij} in T_i. Furthermore, each agent A_k has an associated set of skills $AgentSkills_k$ that A_k is capable of doing, where $AgentSkills_k$ is a subset of S.

Agents use associated skills to complete tasks associated with various jobs, and earn credit by completing all active tasks in a given job. More specifically, each task instance

T_{ij} has an associated $TaskLength_{ij}$, where $1 \leq TaskLength_{ij} \leq \gamma$. To complete task instance T_{ij}, an agent A_k must use an appropriate skill S_i on the task instance for $TaskLength_{ij}$ timesteps. Accordingly, function $C(T_{ij})$ is defined as a value ranging from 0 to $TaskLength_{ij}$, and represents the amount of time that one or more agents have worked on T_{ij}.

To simulate the end results of uncertain information, bounded agent rationality and dynamic, unpredictable environments, jobs in *J* are dynamic and unpredictable. More particularly, task instances T_{ij} in J_{ig} are randomly moved between $ActiveTasks_i$ and $InactiveTasks_i$ on a periodic basis. This may be best understood as a sudden change to a job's solution requirements. For example, despite the best efforts of project management and requirements engineering, software development projects frequently change their required functionality in the middle of development, either due to incorrect understanding of the end-user's original demands or sudden changes demanded by the end user. Accordingly, during the course of such a project, some already-completed portions of the project may become obsolete and new modules may have to be built from scratch. Likewise, in the consulting domain explored below in Chapters 4 and 5, an initial decomposition of a client query may be inaccurate because of misunderstandings with the client, insufficient expertise on the part of the firm in analyzing a novel type of query, or simple environmental dynamics, where a changing external situation demands that different questions be answered.

Figure 4 below summarizes the above environment by presenting an exemplary diagram of agents, skills, jobs and tasks. More specifically, the figure presents three agents, each with a random mix of four skills apiece (out of a total of five possible skills). Likewise, the figure presents two jobs, each with five potential subtasks. In the first job, the first, third, and fifth subtasks are active tasks, and thus must be completed to complete the job, whereas in the second job, only the first and fourth subtasks must be completed. The remaining subtasks are inactive and are thus represented as blocked-out.

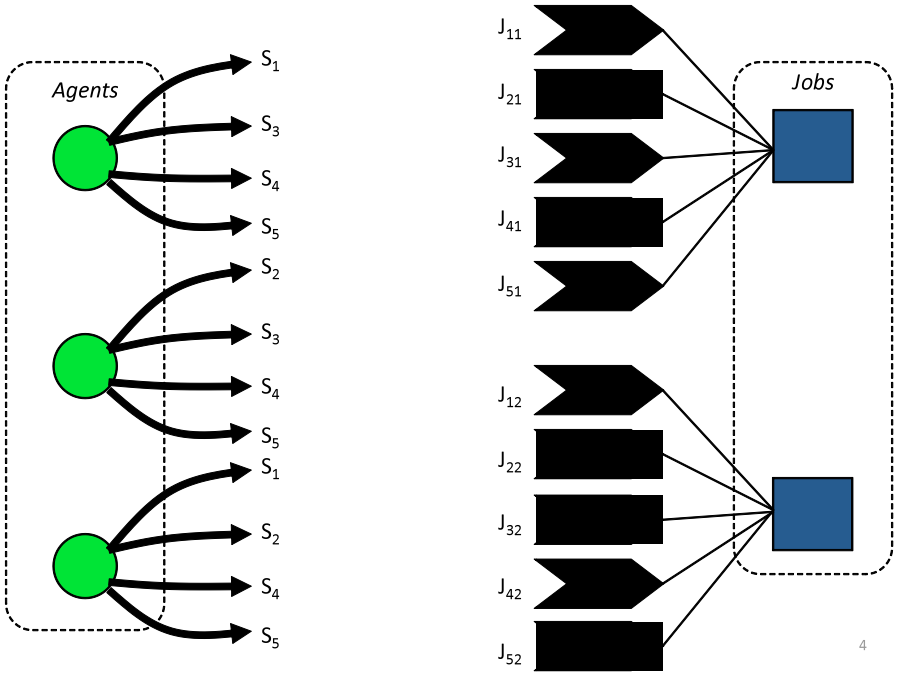


Figure 4: Exemplary jobs and agents

3.3 Simulation Operation

The model described above in section 3.2 has been recreated in a software simulation, which may be used to test agent performance under different environmental conditions as the agent uses different strategies to form teams. Each agent A_k in the simulation has the same number of skills, and each skill in *S* equally common among agents in *A*. Similarly, different tasks are worth the same amount of credit, and agents earn rewards proportional to the TaskLength_{ij} of any task instance T_{ij} they finished. For example, an agent that completed a task over five timesteps earned five credit points, while an agent that completes a task over eight timesteps earned eight credit points. Such assumptions match domains such as collaboration between academic peers in the research community, where the rewards that accrue from completing a task are not monetary but are tied directly to increased respect among peers, or knowledge and skills gained from completing a specific task. It should be noted, however, that many domains will not share these same simulation parameters, and that the experimental simulator could easily be reconfigured

to have some agent skills be more or less common than others, or certain jobs or types of jobs be worth more than others. This will be demonstrated further in Chapters 4 and 5.

Task instances T_{ij} that are moved from $InactiveTasks_j$ to $ActiveTasks_j$ must be started anew, as if no previous work had been done on the now-active task, while only active task instances that have been fully completed are immune from being moved to $InactiveTasks_j$. This decision was based on the reasoning that work that has been fully completed is often used in some way, somehow, whereas partially completed work is often abandoned entirely.

The simulation environment is divided into discrete timesteps, or rounds. During each round, each agent in *A* may coordinate with other agents in *A* to form teams, or, if it is part of a team, may work on a task instance associated with a specific job in *J*. Each agent in *A* can belong to, at most, one team at a time, and each team works on only one job at a time. This is arguably a simplistic assumption, since real world providers of valuable skills or expertise frequently multitask between different projects at the same time. However, many problem solutions require complete focus from the workers involved, or security or other constraints may require exclusivity. Furthermore, requiring each agent to be part of only one team at a time allows us to clearly delineate where an agent is making a contribution. Determining to what degree an agent's partial efforts (or lack thereof) require task reassignments touches on complex multidimensional trust issues (Gujral, DeAngelis et al. 2006), and is out of the scope for this dissertation work.

During each round, an agent A_k may work on a job J_j by utilizing a skill S_i found in $AgentSkills_k$ to work on a task instance T_{ij} found in set $ActiveTasks_j$. Because timesteps are set up to emulate the smallest atomic unit of work that an agent can do with a given skill, each agent utilizes only one skill in any given round. After A_k has worked on T_{ij} for the required number of rounds, T_{ij} is completed.

Credit is distributed to agents when a job J_j is completed, which, in turn, occurs when all task instances in $ActiveTasks_j$ are completed. Upon job completion, credit points for each active, completed task are given to the agent that completed the task. As described above, these credit points are proportional to the length of the completed task (e.g. a completed task of length five would give five credit points). No credit is given for work

on task instances that were moved to $InactiveTasks_j$ before completion, or to agents who worked on, but did not finish, a completed task instance. Accordingly, agents in the simulation may be said to work in a "pay-for-play" environment, where credit is distributed directly to those who have fully completed a given job. The lack of transferable utility between agents also highlights the impossibility of contingency contracting between agents, since agents receive no reward for being associated with a job other than the direct benefit they get from completing subtasks.

Once a job J_{ig} has been completed and paid out its credit, it is removed from J and a new J_j is created and inserted in J. Each new J_j starts with the same number of task instances randomly placed in $ActiveTasks_j$, and task instance in the new job must be completed from scratch.

As described above, the simulation incorporates unpredictability by shuffling task instances between $ActiveTasks_j$ and $InactiveTasks_j$. More particularly, each round a given percentage of jobs in J are randomly shuffled. This percentage is referred to as the dynamicism of the simulation. Each task instance in each selected job J_j has a random chance of being selected for shuffling between $ActiveTasks_j$ and $InactiveTasks_j$, such that, on average, one task instance per selected job is shuffled. However, as described above, task instances that have been fully completed cannot be moved from $ActiveTasks_j$.

3.4 Research Question 1 Approach

Research question 1 states, "what combination of job and team selection heuristics optimizes an agent's utility in a dynamic, contractless environment?" To answer this question, agents in the simulation described above have been divided into multiple classes, each of which executes a strategy consisting of a job selection heuristic and a team selection heuristic. The subsections below describe the communication and coordination mechanisms used by all agents, regardless of class, as well as the heuristics themselves.

3.4.1 Agent Protocol

Agents operating in the simulation environment described above fulfill one of two roles at any given moment. In the first role, an agent takes a proactive "foreman" role and has the responsibility to select an available job to work on, the other agents the foreman will work with, and the specific subtasks that those partners will work on. The foreman communicates these selections to agents via proposal messages which are received by agents in a reactive "worker" role. These worker agents must then decide whether to accept the proposal offers sent out by a foreman agent, or whether to continue with the current job they are working on, if any.

The opportunity to act as a foreman agent is randomly distributed among agents, such that in any given round of the simulation a given percentage of agents will have the opportunity to form teams. Obviously real-world agents in different domains may utilize vastly different criteria to determine when to take the initiative in finding problems and forming teams, including research in agent personality and attitude (Ahn, DeAngelis et al. 2007). Allowing agents to equally and randomly take the opportunity to become foremen is a necessary simplification to focus on the question of *how* teams should be formed, rather than *when* to form them.

The foreman agent then utilizes one of several possible job- and team-selection strategies to determine what jobs to select, what partners to work with, and what subtask each partner should work on. Existing strategies are described further below in section 3.4.2; improving these strategies by creating a comprehensive theory and using agent learning mechanisms is the heart of this dissertation. Accordingly, we remain focused for now on how agents interact with other agents, given the decisions produced by a given strategy.

Once the foreman agent has used its associated team formation strategy to select a job J_w to work on and a potential team $Team_x$, the foreman claims J_w, thereby making it off limits to other teams, and sends proposal messages to potential partners in $Team_x$ indicating the $AssignedInstances_{kw}$ in J_w that a potential partner would work on. Note that, to encourage agents to form teams, $|AgentSkills_k|$ is constant for all agents $\{A_k\}$, and the

initial value of $|ActiveTasks_j| > |AgentSkills_k|$ for all jobs $\{J_j\}$ requiring multiple agents to work together to solve any given job.

When these proposal messages are received, each worker agent ranks the $AssignedInstances_{kw}$ it is currently working on (if any) against one or more proposed $AssignedInstances_{kw}$ using a job selection heuristic associated with that class of agent. If the worker agent finds that its current assigned tasks are preferable to any of the proposals, it continues working on its current job, and the foreman that sent the proposal takes the lack of a response as a decline message. If the agent receives a proposal it finds more attractive than its current job assignment, the agent returns an accept message to the foreman which sent the proposal.

Accordingly, it is noted that agents may stop work on their current assignments at any time upon receiving a more attractive proposal (or, if they become a foreman agents, upon finding a more attractive job to work on). This obviously runs counter to a significant amount of work which his been done in contract negotiation and breaking contracts (Faratin and Klein 2001). However, such work usually involves complete information, and/or occurs between relatively small numbers of agents. In contrast, the scheme described here allows for agents which are better able to take advantage of new opportunities, and better simulates many environments where contracts are largely nonexistent or unenforceable (i.e. informal task forces and many Internet transactions). This lack of commitment between agents, combined with the dynamicism and unpredictability of jobs within the simulation, also makes it desirable for agents to assemble teams that can survive agent defections and changes in the task instances required to finish the job.

If the foreman does not receive accept messages back from all potential partners, the team formation process has failed and the foreman, as well as the agents which accepted the team proposal, must wait for new team proposals or for their next chance to be a foreman. Alternatively, if agents have successfully formed $Team_x$, they begin to work on the task instances associated with J_w. Non-foreman agents may work on J_w until they have completed all task instances in $AssignedInstances_{kw}$. In contrast, the foreman agent must stay with J_w until the job is complete, even if the foreman has completed its assigned

tasks. Once all tasks have been completed, the simulation pays out credit to each agent according to the tasks they have completed.

Agents work on all their assigned tasks in a simultaneous, round robin fashion. For example, if an agent is assigned to tasks one, three, and five, the agent will work on task one for one round, task three the next round, task five the following round, and task one in the fourth round, until all tasks are completed (or existing tasks are removed from the list of required tasks). It is believed that round-robin order gives a better approximation of real-world domains, where agents frequently multi-task their assigned work for a given problem.

While J_w is incomplete, if a non-foreman agent defects from the job, or a new task instance is moved into $ActiveTasks_w$ set, the foreman is responsible for finding an agent to work on the new or abandoned task instance. The foreman may therefore assign the new task instance to the $AssignedInstances_{kw}$ set of itself or a partner agent, in a manner similar to Tambe's work on team reformation strategies (Tambe and Zhang 1998). Previous versions of the simulation have handled this case, wherein no team member has the skill required to handle the new task, in two different ways. In the first, the team simply fails and dissolves with the job uncompleted and no credit earned. In this situation, handling agent defections and the addition of new task requirements is clearly of paramount importance to the agents on a team – if these problems cannot be dealt with, agents will earn no profit.

Alternatively, in other embodiments of the simulation, agent teams have the potential to recruit new members when no existing team members have the necessary skills, using a mechanism similar to the foreman proposals to form new teams described above. Under these rules, the effect of being unable to handle defections and new requirements is potentially less severe, and potentially closer to real-world problem domains, where additional reinforcements are often available. Both of these simulation modes will be utilized in this dissertation to test the relative utility of different heuristic-based strategies, as will be described in further detail below. Figures 5 and 6 below present flowcharts describing the above protocol from, respectively, the perspective of the foreman and the worker agent.

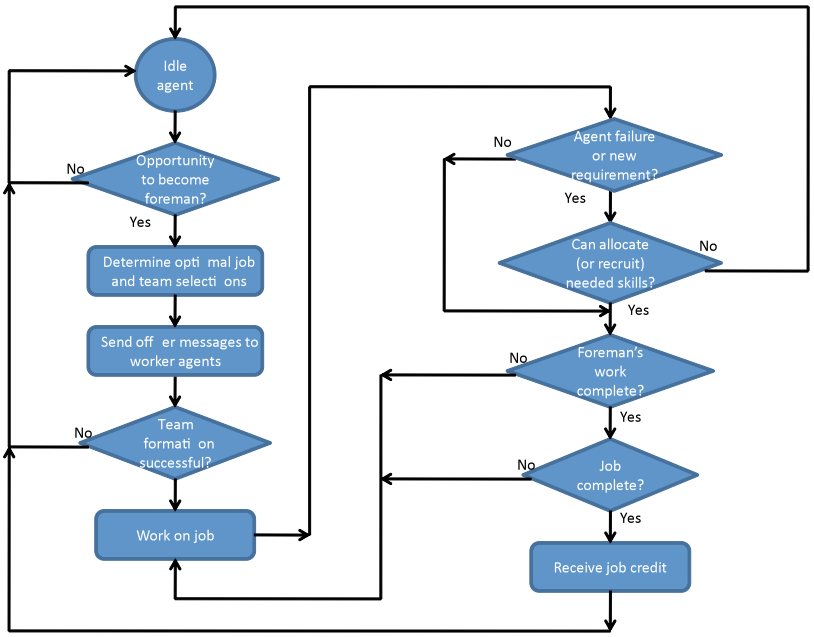


Figure 5: Operational flowchart of Foreman agent behavior

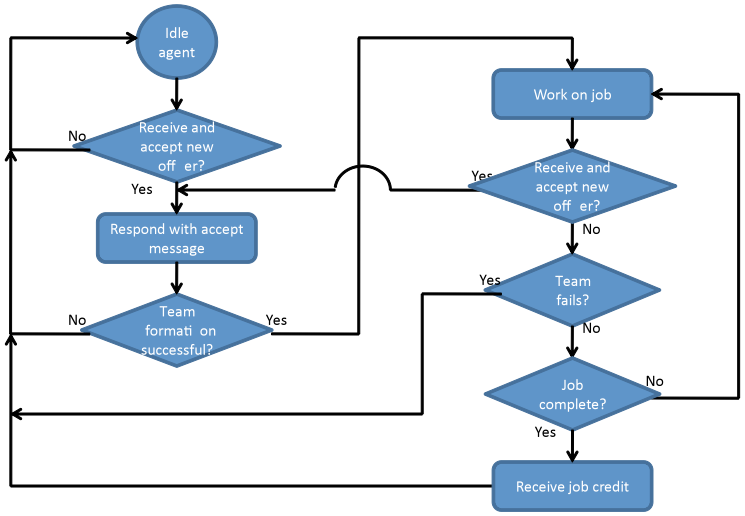


Figure 6: Operational flowchart of Foreman agent behavior

3.4.2 Agent Heuristics

Both worker and foreman agents must make decisions on what jobs to pursue and partners to work with, as described above. Agents in either role utilize paired job and team selection heuristics, described in further detail below, to compare and rank different jobs and potential teams (Jones and Barber 2007). For example, a Greedy job selection heuristic might rank the potential profitability of different jobs based on how much credit an agent could earn by completing assigned tasks in a job, while a Redundant team selection heuristic might rank the robustness of different teams by how many redundant skills each team has to be utilized in case an agent defects from a team.

Agents utilize a mechanism for using both job and team heuristics simultaneously (Jones and Barber 2008). Agents acting in the foreman role initially look at all jobs in J that are not currently being worked on by another team. The foreman utilizes a job selection heuristic to select δ available, top-ranked jobs from J, thereby creating set $P = \{J_w\}$, where $1 \leq w \leq \delta$. For each J_w in P, each foreman agent then generates ε agent teams capable of solving J_w. More specifically, these agent teams each include the foreman agent which generated the team, and one or more other agents A_k, such that the combined skills of all the agents in the team are capable of completing the task instances in $ActiveTasks_w$ of associated job J_w. These teams thus form a set of teams $Q = \{Team_x\}$, where $1 \leq x \leq \delta\varepsilon$.

Once Q is generated each foreman agent uses a selection heuristic R to rank each team in Q, where R is the product of a normalized job selection heuristic and team selection heuristic. R accordingly contains information about both the value of J_w to the foreman agent, via the job selection heuristic, and the potential robustness of $Team_x$, (that is, the team's ability to handle defections and newly added tasks) via the team selection heuristic. Likewise, an agent acting in the worker role uses R to decide between incoming proposals and the job and team it is currently part of, if any.

Each agent A_k in a $Team_x$ is associated with $AssignedInstances_{kw}$, which, after a team has been formed, represents the set of task instances T_{iw} that each agent A_k in the team is assigned to complete in job J_w. Teams are currently assembled via a semi-random approach that seeks to satisfy the various solution requirements one at a time, but nearly

any constraint satisfaction solver could also be used. It is also noted that future work in this field will likely adapt the above mechanism into an "anytime" algorithm (Sandholm and Lesser 1994), wherein agents continually test new configurations of jobs, partners, and task assignments, continually improving their current best selection until some preferred score is reached or external events halt the search process.

Each agent is associated with one job selection heuristic and one team selection heuristic, which are used by the selection mechanisms described above. More particularly, job and team selection heuristics are normalized, so that each heuristic produces a value between 0 and 1 which describes how desirable that job or team is, with 0 being completely undesirable and 1 being completely desirable. Job heuristics attempt to maximize agent profit by selecting attractive jobs, while team heuristics attempt to select teams that are structured to handle sudden changes in the environment or inside the team itself.

The first job selection heuristic is the pre-normalized Greedy heuristic (Eqn. 1) that maximizes the expected reward from a job J_j. While a naive heuristic would simply choose jobs that require the greatest amount of work (and thus the greatest amount of associated reward), the Greedy heuristic takes the dynamicism of the environment into account by giving double weight to portions of a task that have already been completed, thereby giving preferential treatment to large jobs that are less likely to undergo changes before the job is complete. The heuristic is normalized by dividing an agent's potential profit by the number of skills per agent times the maximum length of each task, which represents the maximum theoretical profit an agent can earn from a job.

Greedy heuristic (Eqn. 1): $$\max_{J_j \in J} \sum_{T_{ij} \in AssignedInstances_{kj}} (TaskLength_{ij} + C(T_{ij}))$$

The second job selection heuristic is the normalized Lean heuristic (Eqn. 2) that minimizes the amount of work needed to complete a job, thereby letting agents

opportunistically form teams to quickly solve simpler problems in a mechanism similar to work by Soh and Tsatsoulis (Soh and Tsatsoulis 2002).

Lean heuristic (Eqn 2):
$$\frac{1}{\min_{J_j \in J} \sum_{T_{ij} \in AssignedInstances_{kj}} (TaskLength_{ij} - C(T_{ij}))}$$

Note that these mechanisms stand in contrast to previous work in task selection under uncertain conditions, such as Hannah and Mouaddib (Hanna and Mouaddib 2002), where a problem's uncertain elements are explicitly modeled probabilities. Instead, the heuristics described here operate under any level of uncertainty, from any source. However, future work is possible where the above heuristics are adaptive based on a known or suspected level of uncertainty in the environment, or in a specific problem.

The first team selection heuristic is a Null heuristic that does not rank the teams, but rather keeps teams ordered according to how the strategy's job selection heuristic ranked the jobs associated with each team. This effectively eliminates the team selection heuristic from both selection mechanisms.

The second team selection heuristic is the normalized Fast heuristic (Eqn. 3) that minimizes the maximum amount of work that any member of a $Team_x$ needs to complete. Alternatively, the Fast heuristic could be said to minimize the amount of time needed before the entire team has completed work on associated job J_w.

Fast heuristic (Eqn.3):
$$\max_{Team_x} \frac{1}{\left[\max_{A_k \in Team_x} \sum_{T_{iw} \in AssignedInstances_{kw}} (TaskLength_{iw} - C(T_{iw})) \right]}$$

The third team selection heuristic is the normalized Redundant heuristic (Eqn. 4) that seeks to maximize the number of redundant skills in $Team_x$. In other words, the

Redundant heuristic prefers teams with multiple agents capable of working on active task instances, thereby increasing the ability of a team to deal with the defection of an agent.

Redundant heuristic (Eqn. 4): $$\max_x \frac{\sum_{T_{iw} \in ActiveTasks_w} \begin{cases} 1 < \sum_{A_k \in Team_x} |T_{iw} \cap AgentSkills_k|, & 1 \\ \text{Otherwise}, & 0 \end{cases}}{|ActiveTasks_w|}$$

The fourth team selection heuristic is the normalized Auxiliary heuristic that seeks to maximize the number of auxiliary skills in $Team_x$. In other words, the Auxiliary heuristic (Eqn. 5) tries to maximize the combined skills of a team that are not immediately applicable to task instances in $ActiveTasks_w$, thereby increasing the ability of the team to deal with newly added task instances.

Auxiliary heuristic (Eqn. 5): $$\max_x \frac{\left| InactiveTasks_w \cap \bigcup_{A_k \in Team_x} AgentSkills_k \right|}{|InactiveTasks_w|}$$

Note that, intuitively, the Fast, Redundant, and Auxiliary heuristics each prefer a greater number of partners in a team, since this increases the amount of work that can be done in parallel and the number of unused skills for each partner. Alternatively, the normalized MinPartner heuristic (Eqn. 6) prefers teams with the smallest number of partners, thereby implicitly using a greater number of skills per partner and thus a greater amount of potential profit per partner.

MinPartner heuristic (Eqn. 6): $$\frac{1}{\min |Team_X|}$$

3.5 Research Question 1 Results

As described above, Research Question 1 is addressed by creating an agent simulation that compares the performance of different classes of agents utilizing different job and team selection heuristics. The following results were carried out in accordance with the approach outlined in section 3.4 above, and were published in (Jones and Barber 2007).

Table 1: Experimental parameters for RQ1

Parameter	Value
Number of classes	10
Agents per class	250
Per round chance of agent acting as foreman	1%
Jobs	1000
$\|T\|$	20
$\|AgentSkills\|$	5
Initial size of $\|ActiveTasks\|$	10
Range of TaskLength	1 to 10 rounds
Credit received per round of completed task instance	1
Number of potential teams examined per top-rank job	15
Dynamicism range	0% to 100%, 25% increment
Number of rounds per simulation	2500
Number of simulations per dynamicism step	20

Experiments were conducted using the basic parameters in Table 1, which were selected to broadly model a problem-solving market internal to an organization such as a large corporation or a moderately large number of freelance agents. The experiments tested all strategies against each other simultaneously to see which strategies were most successful in a field of heterogeneous agents. More particularly, a two-factor ANOVA analysis was carried out, and the Fisher LSD method was used to determine significant differences between different strategies at the same dynamicism level.

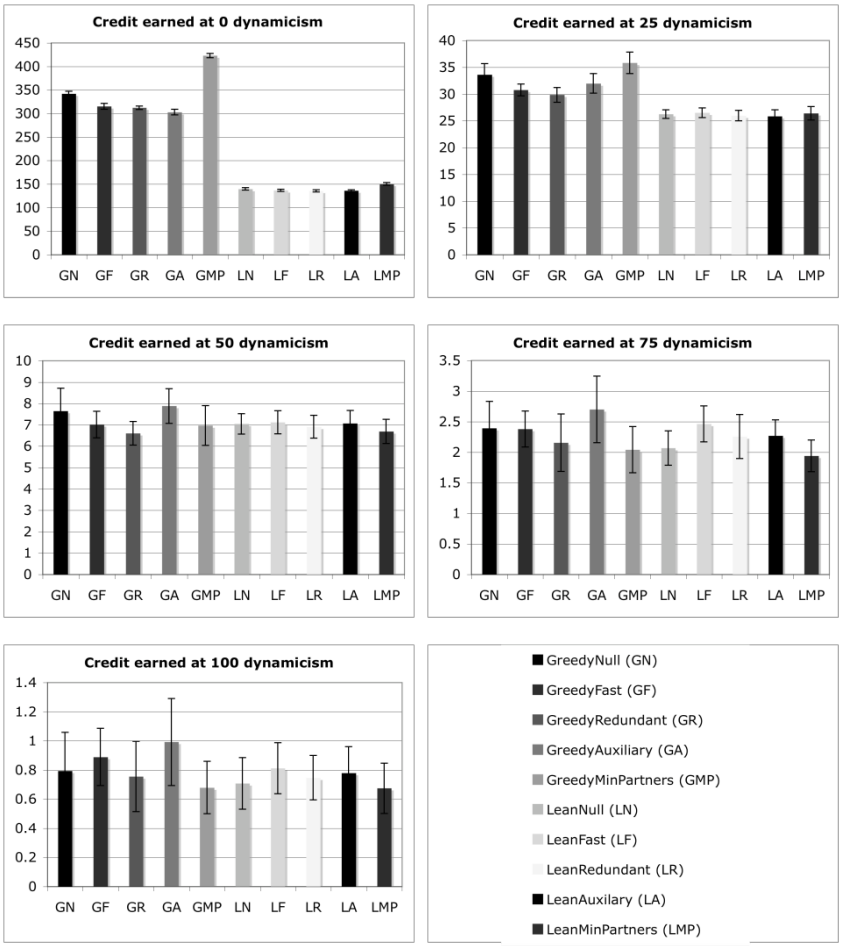


Figure 7: Strategy performance as a function of dynamicism.

Figure 7 displays the average credit earned by each agent class at different levels of dynamicism, along with standard deviation bars for each value. An examination of the data indicates that the credit earned by each class decreased as the level of dynamicism in the environment increased, and that while GreedyMinPartners was the most successful strategy for 0% and 25% dynamicism, GreedyAuxiliary was the most successful strategy for all other dynamicism levels. ANOVA analysis (alpha = .05) indicates that the strategy selection causes statistically significant differences in the results.

Furthermore, Fisher LSD tests (alpha = .05 for all comparisons below) indicate that the top-performing strategy for each dynamicism level was statistically significant from all other strategies, with the exception of 50% dynamicism, where GreedyAuxiliary and GreedyFast were statistically significant from all other strategies but not from each other, and 100% dynamicism, where GreedyAuxiliary was significantly better than all other strategies, save GreedyFast.

Similar results can be seen when examining the aggregate performance of each team selection heuristic, regardless of the job selection heuristic it was paired with: at 0% dynamicism, the MinPartner performed significantly better than all other heuristics, and better than all but the Null heuristic at 25% dynamicism. At 50% dynamicism, the Auxiliary heuristic performed significantly better than all other strategies save the Null heuristic, while at 75%, and 100%, the Auxiliary heuristic performed significantly better than all heuristics but the Fast heuristic. Comparing the aggregate performance of the job selection heuristics, the Greedy job selection heuristic performed significantly better than the Fast heuristic at all dynamicism levels.

To better understand these results, it is helpful to know how successful different classes are at forming teams, as shown in Figure 8 as the ratio between the number of team formation requests by a foreman and the number of teams successfully formed. Furthermore, Figure 9 displays the average success rate of formed teams at completing an assigned job as a function of both agent class and dynamicism level.

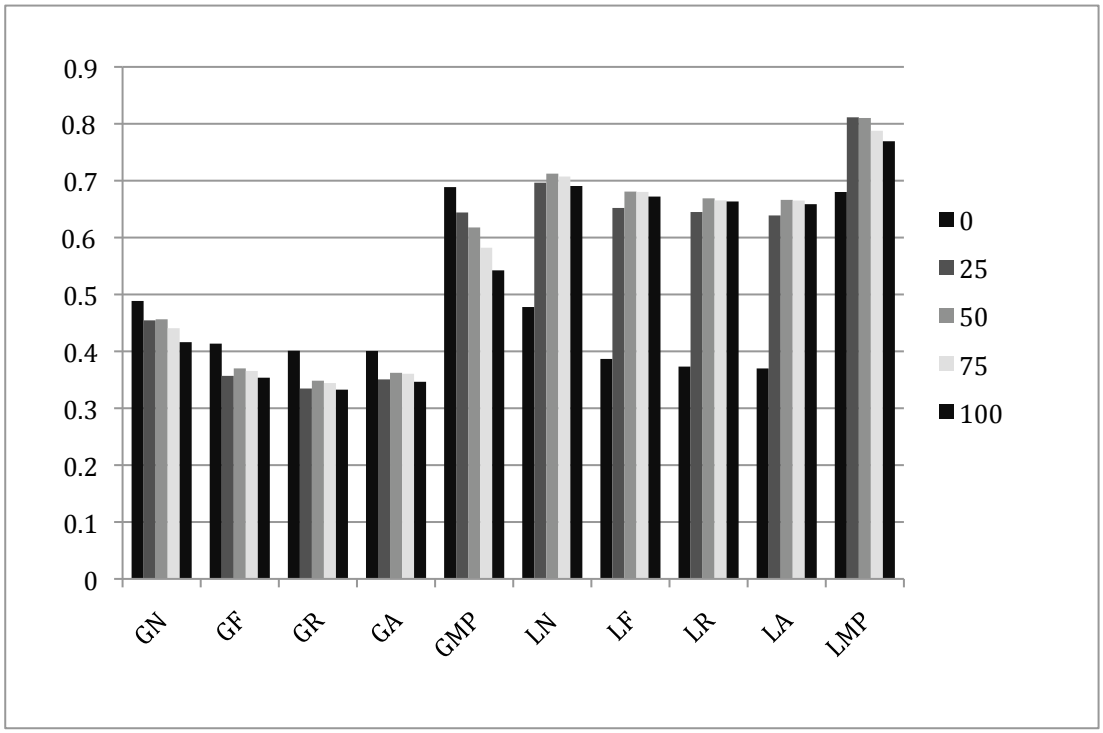


Figure 8: Ratio of teams successfully formed by class and dynamicism level

As can be seen in Figure 8, certain patterns are immediately obvious. Lean job selection strategies are significantly more successful at forming teams than Greedy job selection strategies, except at 0% dynamicism, when job requirements do not change. This may be due to the fact that, in dynamic environments, a subset of jobs will have a lower than average number of tasks, which are more attractive to agents using the Lean job selection heuristic, thereby improving the probability that these agents will join a team.

Furthermore, agents using the MinPartner team selection heuristic are far more likely to successfully form a team. As discussed below, this is likely due in part to the fact that a smaller number of partners imply a smaller number of potential rejections, any one of which can break team formation. In addition, a smaller number of agents means a higher number of tasks assigned to each agent, and accordingly, a greater amount of profit for each agent.

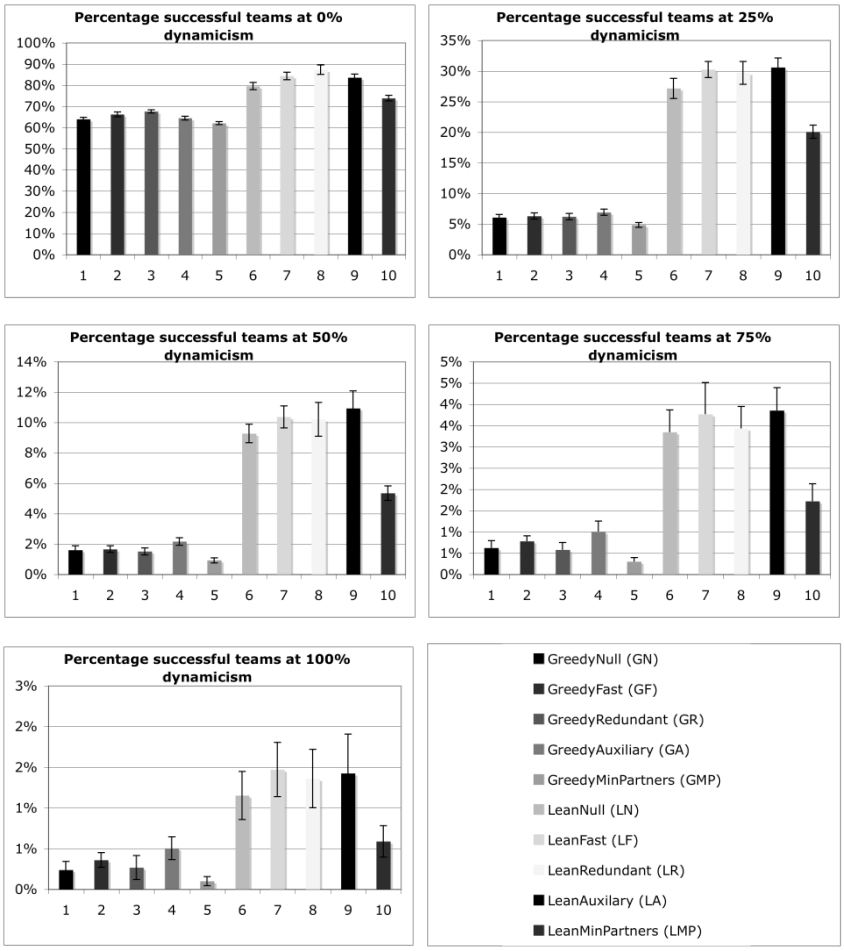


Figure 9: Teams successfully formed by class and dynamicism level

Conversely, Figure 9 shows that, although MinPartners is the most successful at forming teams, it is consistently the least successful heuristic at forming teams which successfully complete their assigned jobs, while the Fast, Redundant, and Auxiliary heuristics are consistently the most successful at forming successful teams. In fact, ANOVA and LSD analysis (alpha = .05) indicates that, although these three heuristics are not always significantly different from each other, they are almost always significantly different than the Null and MinPartner heuristics. Again, as discussed above, this is likely due to the fact that teams formed according to the Fast, Redundant and Auxiliary heuristics are more likely to have team members with unused skills that can be used in

the event that another team member leaves the team, or a new task is added to the list of *ActiveTasks*.

This result suggests that agents that can recognize a team with significant redundant or adaptable characteristics may be more successful than agents that cannot. In other words, agents which can recognize a proposed team that has a lower offered reward but a greater opportunity for success because of its redundant or auxiliary resources may be able to earn more credit.

It is also noted that, although agents using the Lean job selection heuristic are significantly more successful at both forming teams and successfully completing jobs, these agents are generally less successful than agents that use the Greedy job selection heuristic. This may be in part due to the fact that Lean agents generally select jobs with far less work involved, and thus earn far less credit per job. Accordingly, it may be possible to determine under what circumstances a Lean vs. Greedy job selection heuristic is preferred, or even to find an optimal heuristic which combines the two considerations, based on environmental characteristics.

Chapter 4: Expected Utility-based strategy for Robust Team Formation

Research Question 2 states, "How is an agent's utility in a dynamic, contractless environment defined in terms of job and partner selection?" This research question is addressed by first providing a formal, mathematical definition of dynamic and contractless environments in section 4.1, as well as presenting a revised simulation environment which allows for new characteristics, such as quality of service. Based on these definitions, section 4.2 presents mathematical formulas which describe the expected utility for any given agent working with a given team on a given problem. Section 4.3 continues to expand on these results by defining a learning mechanism for an agent to determine the dynamicism of a given environment based on observations made of the agent's immediate environment, thus allowing us to begin to address the first part of Research Question 4, "Can a learning agent adapt its job and partner selections to improve its utility based on observed environmental dynamics?"

4.1 Qualitative Approach For Consulting Domain

Although the modeling approach previously described in section 3.2 is broadly applicable to many different domains, it lacks several useful characteristics that might allow it to be more precisely deployed in domains such as consulting. Accordingly, for reasons which will be described more fully in Chapter 6, section 4.1 gives an overview of certain modifications to our preexisting quantitative model, which now includes the ability to describe quality-of-service demands, different rewards for different types of tasks, and dependencies between tasks. Based on this revised approach, section 4.2 provides a definition and derivation of an expected utility-based strategy for robust team formation in dynamic, contactless environments. Finally, section 4.3 describes how this expected utility-based strategy can be modified to learn, based on observation, environmental variables such as average agent reliability and dynamicism values, and to

adjust the strategy's behavior accordingly. The theoretical work provided in this chapter is then experimentally verified in Chapter 5.

We again begin with a set of jobs, $J = \{j_j\}$, and a set of general tasks, $T = \{t_i\}$. An individual job is composed of a set of specific task instances t_{ij}, so that $j_1 = \{t_{i1}\}$. We also have a set of agents, $A = \{a_k\}$. Each agent has a set of skills $AgentSkills_k$ which is a subset of the set of skills $S = \{s_i\}$. Skills and Tasks correspond with each other, so that skills s_1 is the only skill that can complete an instance $t_{1j.}$

More specifically, each task instance t_{ij} will have a specific quality of service demand associated with it, t_{ij}^q, and each agent skill s_k will have a quality of service capability associated with it, s_k^q, such that a task demanding a given quality of service can only be completed by a skill with an equal or greater quality of service. For example, t_{1j}^2 could be completed by s_1^2or s_1^3, but not by s_1^1. The number of degrees of quality of service is represented by Q.

From a practical standpoint, quality of service might also be thought of as corresponding to the difficulty associated with a specific type of task. For example, in the software domain, creating a small Python search-and-replace script might be thought of as requiring a lower quality of service than creating a larger Java project for parsing web crawling data, which might in turn be thought of as requiring a lower quality of service than creating a distributed operating system in C++. Likewise, service providers capable of providing a given quality of service are usually capable of completing tasks requiring a lower quality of service, but not a higher quality of service. A working engineer with a few years of programming experience, for example, might be ideally suited to create the Java project for parsing web crawling data described above, and also be capable of creating the previously mentioned Python script, but not be capable of creating the C++-based operating system.

Task instances t_{ij} for each job in $\{j_j\}$ are divided into two groups: $ActiveTasks_j$ and $InactiveTasks_j$. All task instances in $ActiveTasks_j$ must be fully completed by an agent before job j_j is considered complete. Task instances in $InactiveTasks_j$ are irrelevant to the completion of the job. For any job j_j, $ActiveTasks_j \cup InactiveTasks_j = \{t_{ij}\}$ for all i, and $ActiveTasks_j \cap InactiveTasks_j = \varnothing$. Alternatively, all inactive task instances may be

thought of as having a required quality of service of level 0, with no payment given for completing the task.

Agents, utilizing skills over time, can complete tasks. Each task instance t_{ij} has a task length TaskLength_{ij} associated with it, which represents the number of timesteps that an agent must use the appropriate skill for the task instance to be complete. Accordingly, function $C(t_{ij})$ is defined as a value ranging from 0 to TaskLength_{ij}, and represents the amount of time that one or more agents have worked on t_{ij}.

Jobs have too many task instances in ActiveTasks for a single agent to complete the entire job by itself; therefore teams of agents are required. For a given job j_w, a team of agents $Team_w$ is formed. $Team_w$ is defined by a set of agents $\{a_n\}$, and a set of assigned task instances $AssignedInstances_{nw}$ for each agent a_n,. Each $AssignedInstances_{nw}$ represents the set of task instances from $ActiveTasks_w$ that a_n is assigned to complete. $AssignedInstances_{nw}$ do not overlap between agents in $Team_w$, and the union of all $AssignedInstances_{nw}$ for all agents in $Team_w$ equals $ActiveTasks_w$.

Agents work on one team, and one job, at a time. To remain part of a team, an agent must work on a task instance from its *AssignedInstances* until all task instances are complete. To leave a team, an agent simply ceases working on its *AssignedInstances*.

Task instances have non-transferable utility, so that task instances pay benefits only when all active tasks in the job are completed, and only to the agent that completed the task instance. (Agents which partially complete a task and quit are therefore not rewarded.) Task payoff depends on three factors – a base payoff per general task type, a length multiplier, and a multiplier depending on the quality of service required for that particular task instance.

So, for example, a general task t_1 might have a base payoff of 3, while a general task t_2 might have a base payoff of 9. To determine the payoff for a specific task instance, we have to know the length of the task and quality of service multiplier for that particular instance. So, for example, if the multiplier is 1 for quality of service level 1, 10 for service level 2, and 100 for service level 3, then $t_{1a}^{\ 1}$ with a length of 3 will have a payoff of 9, $t_{1b}^{\ 2}$ will have a payoff of 90, $t_{1c}^{\ 3}$ will have a payoff of 900. Likewise, $t_{2a}^{\ 1}$ with a length of 5 will have a payoff of 45, $t_{2b}^{\ 2}$ will have a payoff of 450, $t_{2c}^{\ 3}$ will have a payoff

of 4500. Task payoff for a given task instance can be expressed as a function Value(t_{ij}^{q}). Of course, this assumes an agent has the skills necessary to complete the task; if the agent does not, Value(t_{ij}^{q}) is simply 0.

Note that the multiplier differences between task types and tasks lengths given in the example above are approximately linear, while the multiplier differences between different qualities of service are exponential. This is intentional, because although the quantitative model given here and the expected utility-based strategy described in the next section can deal with any arbitrarily different multipliers, it is believed that in many domains, including the consulting domain described in Chapter 6, there are extreme differences in the market power of agents capable of providing different quality of service levels. For example, in the software domain described above, an undergrad engineering student capable of writing Python scripts might be easily hired at a rate of $10/hr, while someone capable of creating an operating system from scratch might be a PhD-level researcher requiring a salary equivalent to many hundreds of dollars an hour.

Task instances may have dependencies between each other, such that one task cannot be completed until another is finished. Task dependencies become active or inactive along with the tasks they are associated with: if task B is dependent on task A and both are initially active, then task B cannot be started until A is completed. However, if task A becomes inactive, than task B can be started immediately. Alternatively, if task A is initially inactive but later becomes active, then work on task B may have to be delayed until task A is complete. This implies that some agents may be relatively idle while they wait for required tasks to be completed before they can start their next task assignment, and that such delays may effect an agent's expected profitability, as an agent is only paid for the tasks it completes.

Given these terms, we can define the following fundamental criteria for this research:

A **Dynamic Environment** is one where task instances are shuffled between $ActiveTasks_j$ and $InactiveTasks_j$, or the quality of service demand of an active task

instance changes with a probability D, where D is the dynamicism of the environment. More particularly, D represents the probability that a specific task instance in a job j_w will either move from Active to Inactive, or vice versa depending on its initial state, or that the quality of service demanded for a specific task instance will change, during the lifetime of the task. In contrast, in a static environment, D is always 0, and the requirements for a job do not change over time. This is described by equations 7 and 8 below.

Environmental Dynamicism (Eqn. 7 and 8):

$$P\left(\begin{array}{l} t_{ij}^{\,q} \in ActiveTasks_{j,time=x}, \\ (t_{ij}^{\,p} \in ActiveTasks_{j,time=(x+Length(t_{ij}))}, q \neq p) \\ (t_{ij}^{\,p} \in InactiveTasks_{j,time=(x+Length(t_{ij}))}) \end{array}\right) = D$$

$$P\left(t_{ij} \in InactiveTasks_{j,time=x}, t_{ij} \in ActiveTasks_{j,time=(x+Length(t_{ij}))}\right) = D$$

An **Unenforcable Contract** between agents is one where agents suffer no penalty for ceasing to work on its assigned task instances. We can therefore write a utility function for an agent with an unenforceable contract, where, upon successful completion of all $ActiveTasks_j$, agents earn a payoff equal to Value(t_{ij}^{q}) of for each task instance they have successfully completed, and the payoff for ceasing to work on *AssignedInstances* is simply 0. In contrast, an enforcable leveled commitment or contingency contract might have a negative payoff for ceasing work on *AssignedInstances*.

Payoff for agents with unenforceable contracts (Eqn. 9):

$$a_k payoff = \begin{cases} \text{if } a_k \in Team_w \text{ for } time_s \text{ to } time_e, \sum Value(AssignedInstaces_{kw}) \\ else, 0 \end{cases}$$

Where $time_s$ is the time the agent joins the team, and $time_e$ is the time all instances in $ActiveTasks_j$ are complete.

A **Selfish Agent** is an agent whose goal is to maximize its own profit, which it tries to achieve by maximizing a heuristic function $H(j_w, Team_w)$ that describes its potential payoff for working on different jobs with different teams. Details of various heuristics are described above in Chapter 3, while the expected utility-based strategy described in section 5.2 below essentially represents another heuristic. For comparative purposes, a selfless agent would have the goal of maximizing the overall profit or utility of an entity other than itself, and accordingly try to maximize a different function, one that represents the overall payoff to a team or the common good. However, all agents described in this dissertation are purely selfish, and only try to maximize their own utility, with no concern for the utility or well-being of other entities.

Selfish Agent maximization function (Eqn. 10):

$$Max\ H(j_w, Team_w)$$

4.2 Expected Utility-based strategy

As described above in section 3.4, previous work in this area has focused on evaluating different heuristic functions used to estimate an agent's payoff for working on

specific jobs and teams. In contrast, the following work tries to build an expected utility-based function that takes into consideration all factors that affect the profit that an agent will actually earn. We therefore present Equation 11, which represents the simple observation that an agent's expected earnings equal its final payoff from a job, multiplied by the probability that the job is completed and the probability that the team is successfully formed in the first place:

(Eqn. 11):

$$ExpectedPayoff(j_w, Team_w) = PComplete(j_w, Team_w) * Payoff(j_w, Team_w) * PFormation(j_w, Team_w)$$

Drilling down, we examine what factors affect the probability that a team will complete a given job. Equation 12 states that a job will be completed as long as the skill set of the team working on the job is sufficient to complete all the active tasks within the job. Given that all teams initially possess the skills needed to complete a given job, there are two ways a team can find itself without a needed skill: either a new task is added to the list of active tasks that the team lacks the ability to handle, or a current team member quits, depriving the team of a previously needed task. (This analysis obviously applies only to the case where new team members cannot be recruited.)

(Eqn. 12):

$$PComplete(j_w, Team_w) = 1 - (P(Unsolvable(j_w, Team_w) \cup Defect(j_w, Team_w)))$$

To determine probability that a new task will be added to a job that the current team cannot handle, we must determine the probability that a task will transition to a specific alternate quality of service state. Since the probability that a task will transition at all is D, and there are Q other states to transition to (inactive tasks can transition to 1 of Q quality of service levels, active tasks can transition to (Q-1) alternative quality of service levels, plus one inactive state) then the probability of it transitioning to a particular state

is D/Q. Depending on the skills composition of the team, some quality of service levels can be handled and some cannot; for each task we count the number of quality of service states that cannot be handled. If we represent the number of states per task that cannot be handled by X, the probability that a specific task will transition to an unsolvable state is XD/Q. We can then write the probability that this transition will not occur as 1-(XD/Q), and multiply this value for all tasks together to determine the probability that no task in the job will transition to an unsolvable state. The probability that the job *will* transition to an unsolvable state is then 1 minus this latter number, as shown in Equation 13.

(Eqn. 13):

$$PUnsolvable(j_w, Team_w) = 1 - \prod_{i \in S} (1 - (\frac{D}{Q}(\bigcup AgentSkills_w \cap s_i)))$$

The probability that a current team member will defect from the current team depends on multiple factors, chief of which is the logic driving the agent's decision process for leaving a team. In a massively multiscale environment composed of extremely heterogeneous agents, we might plausibly make the argument that because there are so many possible mechanisms for agents to use in deciding when to defect, some of which may not even be rational, it is therefore impossible to actually calculate the probability that an agent will defect from the team.

For example, consider two plausible heuristics that an agent might use in ranking potential jobs. A greedy heuristic might simply seek to have each agent earn more from a job – thus, if agent earnings are roughly proportional to how much work an agent does, then a greedy agent will prefer a job where it is given more to do, and will conceivably leave its current job if another job with substantially more work/earnings involved, is offered to it. Alternatively a "lean" heuristic is one where an agent seeks only to work on many quick, low-risk jobs, rather than a smaller number of large, high-risk jobs. Again, if we assume that agents earn roughly in proportion to the amount of work they do, then

"lean" agents will prefer smaller jobs, and conceivably leave their current job if a job which can be completed more quickly is offered to it.

Note, however, that these approaches are directly opposed to each other – a new job which would tempt an agent using a greedy strategy to defect will not tempt an agent using a lean strategy, and vice versa. In the instance where agent behavior is relatively heterogeneous, we might be able to develop a theoretical model that derives an agent's chance of defection based on how happy it is with its current job, and how likely it is that a "better" offer would come along. However, among heterogeneous agents, such a model is impossible because of conflicting heuristics, and we are left with the option of simply learning by observation, over time, how likely an agent is to defect. This could be done by direct experience, shared reputation, or a mixture of both, either for every individual potential partner agent, for different classes of similar agents, or for a general profile of all possible agents.

We can therefore describe a probability PAgentDefect, which is the probability that an average agent will defect. From this we can define PDefect(j_w, $Team_w$) by first figuring out how many partner agents are critical to the team (i.e. how many agents provide a required skill that no other agent on the team can provide). We can then calculate the chance that no such critical defection occurs, as shown in Equations 14 and 15.

(Eqn. 14):

$$CriticalAgents = \sum_{A_k \in Team_w} \begin{cases} \text{if } (|ActiveTasks_w| > |\bigcup AgentSkills_{Team_w} - AgentSkills_{A_k}|), 1 \\ \text{else, } 0 \end{cases}$$

(Eqn. 15): $$PDefect(j_w, Team_w) = 1 - (1 - PAgentDefect)^{CriticalAgents}$$

Addendum: future work will reexamine equations 14 and 15 based on the premise that simultaneous defections by agents are possible. However, given that altering the equations above to include team failures due to simultaneous defections would make the EU strategy more pessimistic, and the results of section 5.2 below suggest the EU

strategy is already overly pessimistic, this is not currently believed to be a fatal flaw in the EU strategy.

Note that we may wish to treat the probability of an agent defecting and the probability of a problem becoming unsolvable as independent, since there are potentially an exponential number of sub-teams that can be formed once agents start defecting, each of which has a different likelihood that it will encounter an unsolvable job configuration. Under this simplifying assumption, Equation 16, which describes completion probability, then becomes:

(Eqn. 16):
$$PComplete(j_w, Team_w) = 1 - (PUnsolvable(j_w, Team_w) + PDefect(j_w, Team_w) - (PUnsolvable(j_w, Team_w) * PDefect(j_w, Team_w)))$$

The probability that a team will successfully be formed is the product of how likely each potential team member (less the agent running the expected utility strategy) is to join the team. This likelihood in turn is the sum of likelihood that a team member is idle (and thus, completely likely to join a proposed team) and, in the case that the agents are not idle, the likelihood that they will defect from their current teams. Equation 17 therefore describes the probability that a team will successfully form:

(Eqn. 17):
$$PFormation(j_w, Team_w) = (PIdle + PAgentDefect(1 - PIdle))^{(|Team_x| - 1)}$$

The payoff each agent on the team receives, assuming the job is completed, is the sum of how many time steps each agent has spent working on completed tasks. To a first approximation, this value is the sum of three distinct components: the payoff value of all the tasks the agent is currently assigned to, the payoff value of all the tasks that the agent may receive, should they become active, and the payoff value off all tasks reassigned to the agent because another agent on the team defects.

(Eqn. 18):
$$Payoff(j_w, Team_w) = ActiveTaskPayoff(j_w, Team_w) + InactiveTaskPayoff(j_w, Team_w) + ReassignedTaskPayoff(j_w, Team_w)$$

We begin by calculating the expected value of a single task, ExpectedValue(t_{ij}^{q}). We calculate this by adding the value of every possible quality of service, multiplied by the probability that that quality of service will be demanded.

(Eqn. 19):
$$ExpectedValue(t_{ij}^{q}) = \sum_{n=1..|Q|} \begin{cases} \text{if } n = q, (1-D)Value(t_{ij}^{q}) \\ \text{else, } \frac{D}{Q} Value(t_{ij}^{n}) \end{cases}$$

By calculating this for all active tasks an agent is assigned to, we can figure out the first part of the equation above.

(Eqn. 20):
$$ActiveTaskPayoff(j_w, Team_w) = \sum_{AssignedInstances_{kw}} ExpectedVdue(t_{ij}^{q})$$

The expected payoff for all inactive tasks that an agent may be assigned is the probability that the task will become active multiplied by the probability that the task will be assigned to a specific agent, multiplied by the expected payoff for the task. (The work below assumes that an inactive task (quality of service level 0) can become active with any quality of service level, and that different subsets of agents may be capable of handling each different quality of service level task instance.)

(Eqn. 21):
$$InactiveTaskPayoff(j_w, Team_w) = \sum_{t_i \in InactiveTasks_w} \sum_{n=1..Q} \frac{D}{Q} \frac{ExpectedValue(t_{ij}^{n})}{\left| \sum_{A_k \in Team_w} (AgentSkills_k \cap t_{ij}^{n}) \right|}$$

Finally, the expected payoff for active tasks reassigned from a defecting agent is the probability that another agent will defect, multiplied by the probability that the task will

be assigned to a specific agent, multiplied by the expected payoff for completing that task.

(Eqn. 22):

$$ReassignedTaskPayoff(j_w, Team_w) = \sum_{ActiveTasks_w - AssignedInstances_{kw}} \frac{ExpectedValue(t_{ij}^{\ q})PDefect}{\left| \sum_{A_k \in Team_w} (AgentSkills_k \cap t_{ij}^{\ q}) \right|}$$

Although we have determined the likely payoff for a given job/team combination, we have not yet determined the likely duration of such a job. Calculating the duration of such a job is extremely difficult due to several factors – changing tasks, changing dependencies, and a changing team of agents to complete the same work. Rather than explicitly identify every possible permutation of teams and task assignments, we instead take a more probabilistic approach and choose to represent the range of potential job durations as being part of a distribution range – more specifically, a beta distribution, using methodology borrowed from the project management domain.

More specifically, a beta distribution is useful because we first expect that the duration distribution will be asymmetrical – that defections and additional tasks added to the job will lengthen the duration, but, because of existing dependencies, removing existing tasks may not significantly shorten the duration. (E.g., if tasks A and B are being worked on in parallel, and are prerequisites for task C, then the removal of task A or B by itself will not greatly shorten the expected duration.)

A beta distribution allows us to model this asymmetry by allowing us to calculate an expectation value for the duration based on 3 easily calculable numbers: the most likely duration *m* (i.e. the starting duration, or simply the critical path of the original task assignments), the minimum duration *a* (i.e. 0, assuming all currently assigned tasks drop out) and the maximum duration *b* (i.e. the worst-case scenario, assuming all possible tasks are assigned to the agent in question, and all possible dependencies are enabled, and worked as slowly as possible.) Note also that the maximum duration *b* is further refined

as work on a job progresses and some tasks become permanently inactive, and thus not part of the maximum possible duration. Once we have these three values, Equation 23 describes our expected duration value, based on techniques taken from project management methodology (Shtub, Bard et al. 2005).

(Eqn. 23): $Duration(j_w, Team_w) = \frac{a + 4m + b}{6}$

Furthermore, by dividing our expected payoff value by this number, we can figure out the expected earnings per unit of time for a given job. It is this number which takes the place of H in Equation 10, since we wish to maximize an agent's earnings within a fixed timeframe, rather than simply maximizing total earnings across an infinite timeframe.

(Eqn. 24): $ExpectedEarningsPerTimestep(j_w, Team_w) = \frac{ExpectedPayoff(j_w, Team_w)}{Duration(j_w, Team_w)}$

4.3 Learning modifications to the Expected Utility-based strategy

The expected utility-based strategy described in section 4.2 above is capable of deciding which combinations of jobs to work on and agents to work with are likely to be most profitable. However, the above strategy does depend on certain known quantities such as PAgentDefect, PIdle, and the dynamicism of the environment, D. For comparison purposes with existing heuristic-based strategies, these values are provided to the strategy in many of the experiments carried out in section 5.1 below.

However, in several real-world domains these values cannot be directly provided to the strategy because they are not directly known, but rather must be learned by the strategy itself. Accordingly, this section describes several learning mechanisms which allow an agent executing the expected utility-based strategy to be placed into an environment where it does not know the local dynamicism or agent reliability, learn the

values, and thereby improve its earnings performance, thereby allowing us to address RQ4.

Initial approaches to the problem of learning values such as environmental dynamicism focused on reinforcement learning mechanisms such as Monte Carlo or Dynamic Programming methods (Sutton and Barto 1998). However, upon further reflection reinforcement learning is not necessary for the learning required in this instance. Specifically, reinforcement learning is most useful when learning how to deal with a problem that cannot be easily or fully modeled by direct observation, and often involves a direct mapping of specific actions or strategies to expected numeric rewards based solely on repeated experience.

Moreover, reinforcement learning mechanisms observe and learn specific reward distributions that are unique to a given action. This differentiation of observed values is not necessary in cases where the observed value distributions are independent of the action taken. Indeed, this is the case with regard to the dynamic, contractless domain examined by this research, since key variables such as the dynamicism of the environment are fully independent of the actions (e.g. selecting jobs and partners) taken by the agent.

Accordingly, the focus of this section is primarily on how to accurately translate observations of events such as active/inactive status changes of task instances to meaningful values such as the dynamicism value D. In fact, in the case of variables such as PIdle and PAgentDefect, little translation is necessary.

One of the simplest mechanisms for estimating PIdle is for the agent executing the expected utility-based strategy to keep track of its own percentage of time spent idle vs. time spent working as a worker or foreman agent. This value can then be extrapolated to be representative of all agents, and is a reasonable assumption to make in instances where all agents take the initiative to be leaders approximately the same percentage of the time, and where the primary difference between agents is their skill set, rather than their strategy (which cannot be observed by individual agents in these experiments.) This is the mechanism used for the experiments carried out below in chapter 5.

However, it is noted that, should these assumptions not hold true, a different mechanism might be required for determining the probability of team formation than the one described in Equation 17 above. Such an alternate mechanism might involve directly observing the percentage of times a team formation attempt is successful, or even using reinforcement learning to distinguish different probabilities of successful team formation based on how many potential partners are in a given team.

Likewise, PAgentDefect may be directly observed by simply keeping track of how many agents leave a team in the course of working on a job, and dividing this number by the total number of agents worked with. This is the mechanism used in the experiments described below in Chapter 5. More complex methods are also of course applicable (Fullam 2007), but such methods are more properly thought of as belonging to the trust and reputation research domain, and will not be further discussed here.

In contrast, determining the dynamicism value D purely from observations of how frequently task instances change is somewhat complicated. If all jobs ran to completion regardless of task changes, it would be trivially easy to find on average of how many task instances changed inactive/active state, or changed the quality of service required for completion, and calculate D accordingly. However, because changes in job requirements frequently make a job unsolvable for a given team, jobs and individual task instances are only partially observed. This in turn means that not all examples of status changes for a given job will be counted, which makes any such average derived from these observations inaccurate.

Furthermore, although changes, once they occur, can be considered fully observed, task instances which do *not* change cannot be considered fully observed until the task is fully complete, which may not occur if the team disbands prematurely. If a task instance has been 80% completed and has not changed, should it be counted as being an unchanged task or not?

Lastly, it is also the case that task instances do not change from one state to another in a uniform and consistent manner. More specifically, task instances which change from inactive to active occur within the first 10 time steps of a job's lifetime, whereas task instances which are already active and change quality of service requirements, or change

to inactive task instances, only occur once the task instance has been partially completed to some preset degree. (See section 3.3 above.) This may lead to situations where many task instance changes are observed soon after a job is started, whereas the vast majority of unchanged task instances cannot be observed until a substantial portion of the job is completed, again leading to inaccurate estimates of the dynamicism level.

Accordingly, the equations below describe the mechanism by which dynamicism is observed. Broadly speaking, we wish to determine the percentage of task instances which change state, which we can calculate by dividing the number of observed task instance change events by the number of total task instances observed. Determining the number of observed task instance changes (*ObservedActiveEvents*) is trivial – all we have to do is count the number of task instances where the quality has changed during the lifetime of the job, as shown in Equation 25.

(Eqn. 25): $ObservedActiveEvents(j_w) = \sum_{t_{iw}{}^q \in j_w} \begin{cases} \text{if } q \text{ is constant, } 0 \\ \text{else, } 1 \end{cases}$

Determining the number of observed task instances that do not change (*ObservedStaticEvents*) is more difficult. As discussed above, task instance event may be partially completed when a job is abandoned, and if the task instance event has not changed state during that partial observation, that does not imply that a static event (that is, a lack of change) has occurred. Rather, at best we can talk about the probability that a static event has occurred. Moreover, since the probability that a task instance will change is evenly distributed throughout its lifetime (i.e. a task instance can change quality of service or become inactive after each timestep that it has been worked on, with the exception of the last), the probability that a static event has occurred is proportional to the task instance's completion percentage. For example, if 9 out of 10 total steps of a task instance have been completed, there is a 90% chance that the task instance is static. We can therefore talk about observing *partial* static events, and add these partial static events together to determine the total number of static events observed thus far in a job.

Further note that these partial static events are calculated differently depending on whether the task instance is initially active or inactive. As described above, inactive tasks that change will do so towards the beginning of a job's lifetime, within the maximum length of an individual task instance (e.g. 10 timesteps). Alternatively, active task instances which change do so depending on how much of the instance has been completed. Accordingly, Equation 26 calculates *ObservedStaticEvents* differently depending on whether the initial quality of service is 0 (inactive) or not.

(Eqn. 26):

$$ObservedActiveEvents(j_w) = \sum_{t_{iw}{}^q \in j_w} \begin{cases} \text{if } q \text{ is not constant, } 0 \\ \text{else if initial q } = 0, \left(\dfrac{CurrentTime \ - \ StartTime}{MaxTaskLength} \right)^{ObservationExponent} \\ else, \left(\dfrac{C(t_{iw}{}^q)}{Length(t_{iw}{}^q)} \right) \end{cases}$$

Although observing partial static events allows us to better estimate the dynamicism value for an environment, it can also be misleading in some circumstances. More specifically, at very high levels of dynamicism (i.e. 60% or greater), teams have a much higher chance of failure, and as such disband very quickly. The observations made during these brief intervals of job activity may not accurately capture the true levels of dynamicism, since many partial static observations may indicate that task instances are not changing, although these same instances would change if given enough time.

For example, consider a job comprising 10 active task instances, where all task instances will eventually change. Assume that work begins simultaneously on many different task instances, but only a small percentage of the instances are complete before some of the task instances change to a higher quality of service that the team does not possess, forcing the team to disband. In this situation an agent might have observed 3 out of 10 active events, and might have observed 2 partial static events in sum. The

straightforward calculation of dynamicism from this data might lead the agent to believe that D = 3 / (2+3), or 60%. However, in actuality, since all task instances would have eventually changed, the correct value of D is 100%.

Accordingly, it is the case that, at higher levels of dynamicism, partially observed static events should count for less. We accomplish this by weighting these events differently if the observed dynamicism is higher. Specifically, we wish to raise the value of a partially observed static event to an exponential power of 1 or greater, thereby giving less importance to largely incomplete task instances, while maintaining the importance of mostly or fully complete task instances. Equation 27 provides a model for how the *ObservationExponent* value seen in Equation 26 would be calculated, whereby at observed dynamicism values of less than .58, the value is 1, at observed dynamicism values of greater than .68, the value is 2, and between .58 and .68, the value increases linearly.[1] Therefore, if the observed dynamicism value is 80%, then a task instance which had been 20% completed but not observed to change would count as only .04 of a static event, while a task instance that was 90% complete would count as .81 of a static event, and a fully completed task instance that had not changed would count as 1.00 static events.

(Eqn. 27): $$ObservationExponent = \begin{cases} D < .58, 1 \\ .58 \le D \le .68, 1 + (D - .58) * 10 \\ D \ge .68, 2 \end{cases}$$

Finally, Equation 28 shows how the values of *ObservedActiveEvents* and *ObservedStaticEvents* are brought together to determine the observed dynamicism value for a single job – simply put, the number of observed active events is divided by the number of total observed events, which is in turn the sum of the observed active events and observed static events.

[1] The value .58 was determined through informal experimentation, and gives reasonably accurate results, as seen below in chapter 5. However, it is freely admitted that this method is only a first approximation, and that further refinement as to how *ObservationExponent* is calculated could lead to better approximations of the true value of D.

(Eqn. 28): $$ObservedDynamicism(j_w) = \frac{ObservedActiveEvents(j_w)}{ObservedActiveEvents(j_w) + ObservedStaticEvents(j_w)}$$

Equation 28 gives the observed dynamicism for a single job; to estimate the value of dynamicism for all the jobs in an environment, we simply average our observations for multiple observed jobs, which allows us to converge on the actual value of D, assuming D stays fixed. Moreover, basic statistics tells us that the more observations we make of D, the closer our average gets to the actual value of D.

However, if D is not fixed, but rather changes over time, past observations of D might need to be discounted. Two straightforward ways of doing this are to create a sliding observation window, where only relatively recent observations of D are used to calculate the probable value of D, or to exponentially discount the influence an observation of D has on the average depending on the age of the observation. As will be seen below in Chapter 5, both of these mechanisms are compared to the all-inclusive running average of D described above to test the efficacy of each mechanism against a changing or outright random value of D.

Chapter 5: Experimental Results

The expected utility-based theory described above in Chapter 4 is easily translatable to an expected utility (EU) strategy which is believed to be more profitable than existing heuristic strategies in dynamic, contractless environments. Accordingly, this chapter presents the results of several experiments that examine the effectiveness of the expected utility-based (EU) strategy, both in comparison to new and existing strategies, and in terms of how accurate the strategy's predictions of future job behavior are. This work thereby allows us to address Research Question 3, "How do sagents acting in accordance with the utility theory found in RQ2 perform in contractless, dynamic environments?"

More particularly, section 5.1 compares the profitability and effectiveness of the expected utility-based strategy (EU strategy) against both the existing heuristic-based strategies described in Chapter 3, and several new strategies which represent an absolute standard of comparison – a "Master" strategy which prohibits partner agents from quitting a team, and an "Oracle" strategy which is aware of all task instance changes a job will undergo before the job begins. Section 5.2 examines the accuracy of the individual components of the expected utility-based (EU) strategy calculation to determine how those predictions (i.e. expected profit, job duration, and probability of success) compare to actual observed results.

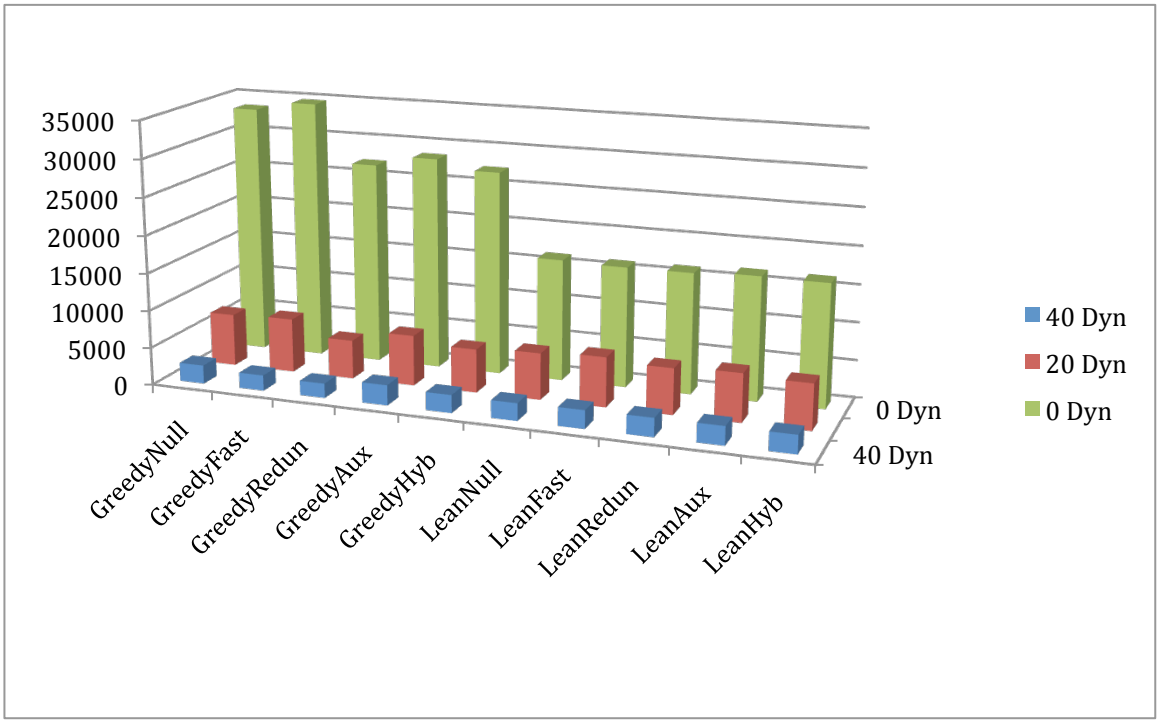
Lastly, we finish addressing RQ4 in section 5.3 by examining the different learning mechanisms described above in section 4.3 (observing dynamicism per job and determining environmental dynamicism, D, via a running average, sliding window, or exponential weighting of past data) by first comparing the performance of each learning mechanism to the performance of an expected utility-based strategy which is provided the exact value of D. The section then examines which mechanism is most profitable when faced with a simulation environment where D changes over time, and how closely the internal estimates of D track the changing actual value of D in the environment.

5.1 Performance of Expected Utility-based strategy

We begin with a brief review of the relative performance of some of the heuristic strategies described above in Chapter 3. More particularly, Figure 10 below illustrates the relative performance of ten different heuristic-based strategies using the parameters found in Table 2. Note that agent classes and skills are stochastically assigned to agents, so that, on average, agents and skills are evenly distributed between classes. However, the resulting spread in exactly how many agents and skills are found in each simulation run is likely decrease the sensitivity of the results presented below to any specific simulation parameters. While the Greedy and Lean job selection heuristics and the Null, Fast, Auxiliary and Redundant team selection heuristics are identical to those described above in Chapter 3, the Hybrid team selection heuristic is new. Simply put, the Hybrid team heuristic is a straightforward average of the Auxiliary and Redundant team selection heuristic.

Table 2: Experimental parameters for Figure 10

Parameter	Value
Number of agents	2000
Per round chance of agent acting as foreman	5%
Jobs	1000
Number of total skills	10
Degrees of quality of service per task	3
Number of skills per agent	5
Average number of active tasks per job	6
Range of task lengths	2 to 10 rounds
Range of task values	1 to 10
Value of quality of service multiplier	1, 10, or 100
Average probability a task has at least one prerequisite task	50%
Number of top-rank jobs examined	5
Number of potential teams examined per top-rank job	25
Dynamicism range	0% to 100%, 20% increment
Number of rounds per simulation	1000
Number of simulations per dynamicism step	10



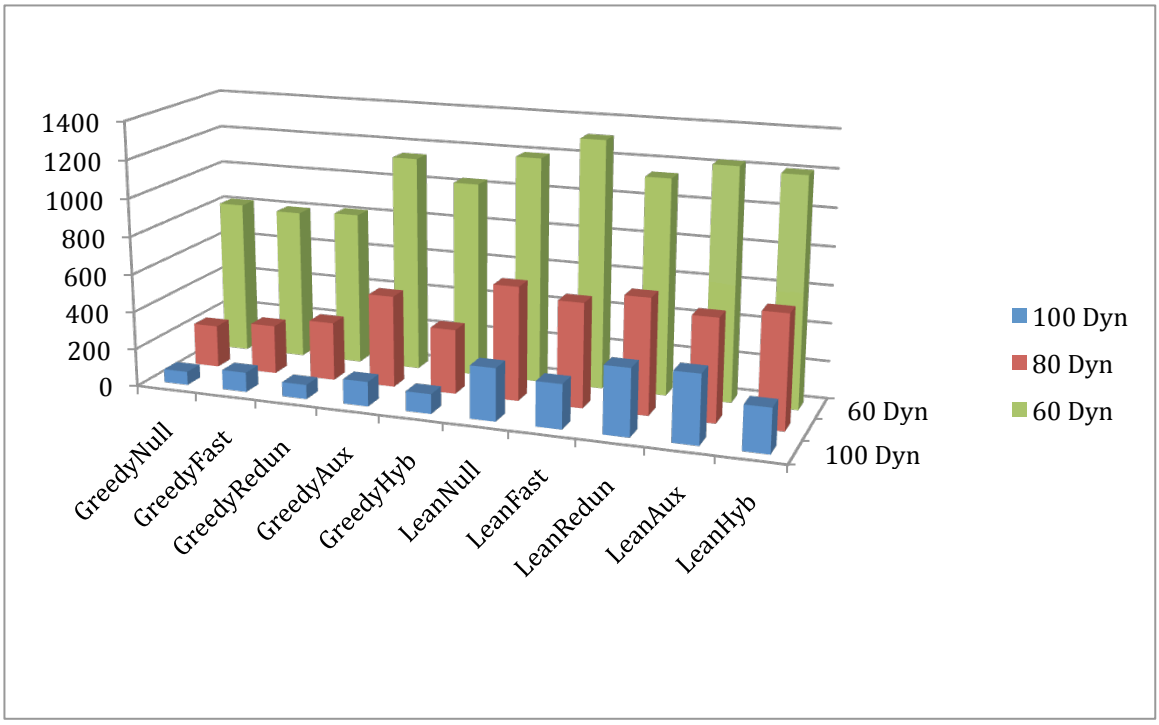


Figure 10 a-b: Average total profit by class and dynamicism level for all heuristic-based strategies

As can be seen from the graphs above, agents executing the greedy job selection heuristic earn more profit when jobs are static (i.e. D = 0%), but as dynamicism increases, agents executing the lean job selection heuristic earn more profit. T-testing of the two populations confirms a statistically significant advantage of greedy job selection heuristics in static environments, and an advantage for lean job selection heuristics in environments with 60% or greater dynamicism.[2] No specific team selection heuristic was notably superior across the entire dynamic range. It is noted that the differences in scenario details makes a direct comparison between these results and those given in chapter 3 pointless (e.g. no quality of service differences in earlier results, different mechanism used to determine agent profit), although these graphs are useful for comparison purposes when considering relative performance of other strategies (e.g. expected utility-based or oracle strategies) in combination with these heuristic-based strategies.

As previously discussed above, in addition to comparing the expected utility-based strategy described in Chapter 4 to existing heuristic-based strategies, we also wish to compare the expected utility-based strategy to a more absolute measure of profitability. Towards that end, we have created three additional strategies which do not function under the same set of rules previously established for our existing strategies. More specifically, we have created a **Master** strategy, which functions identically to a GreedyNull heuristic strategy, except that any partner agents working on a team with the Master agent as the foremen are prohibited from quitting the team unless they have completed all their assigned tasks.

Similarly, we have created an **Oracle** strategy, which again functions the same as a GreedyNull heuristic strategy, except that the Oracle strategy is capable of perceiving both the current active/inactive status and quality of service status of each task instance, and what the ultimate status of that task instance will be. In other words, an Oracle agent perceives both the current and final skill requirements of a job, and as such, will

[2] Note that all statistical comparisons designated as significant feature alpha = 5%.

preferentially select teams that it knows can complete all the current and future tasks of a job over teams that cannot. This gives the Oracle agent a far better chance of completing jobs as a foreman, even in highly dynamic environments.

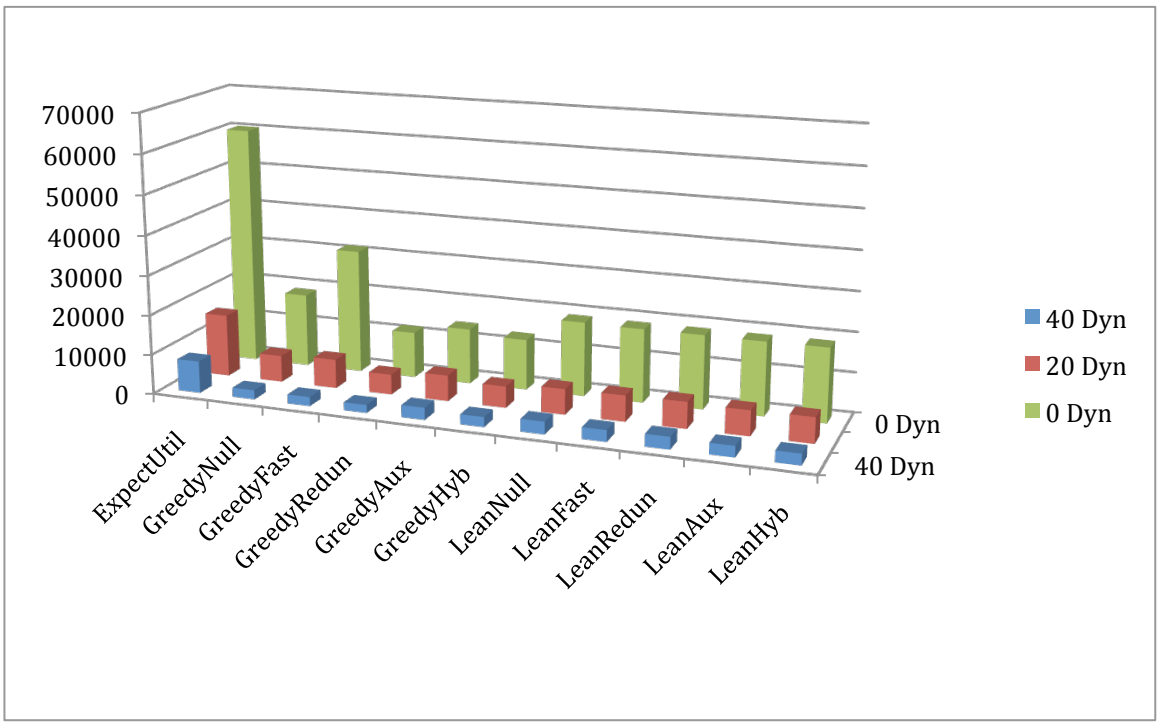
Lastly, we have combined the Master and Oracle strategy to create a **MasterOracle** strategy, which is again similar to a GreedyNull strategy, except that like the Oracle strategy it preferentially selects teams that it knows can successfully complete a job, and like the Master strategy, partner agents cannot quit a team that is headed by a MasterOracle agent. Unlike the expected utility-based strategy, therefore, the MasterOracle strategy operates in a static, contracted environment, since it does not risk agent defections or unexpected changes in task requirements for a job. The MasterOracle thus represents a best case performance for an agent operating in a dynamic, contractless environment, and is therefore a more objective standard of comparison than the existing heuristic-based strategies.

Figures 11 shows the performance of the same heuristic-based strategies from Figure 10, but with the addition of the EU strategy. More specifically, the experimental parameters laid out in Table 2 were used for the following experiments, except that rather than 2000 agents being assigned into 10 classes, the 2000 agents were assigned into 11 classes. As can be readily seen from Figure 11, the expected utility-based strategy significantly outperformed all other heuristic-based strategies, often earning more than double the next-highest earner. ANOVA and t-testing confirms both a statistically significant spread in strategies, and a statistically significant advantage of the EU strategy over all other heuristic-based strategies in all dynamicism levels.

Interestingly, agents executing lean job selection heuristics earned more than agents executing greedy job selection heuristics at all dynamicism levels, as opposed to Figure 10, where agents using the greedy heuristics earned more in static environments. Figures 12a and 12b, which show the breakdown of average profits earned in the worker role as broken down by class and dynamicism, indicate that this is likely because workers executing the lean job selection heuristic earned significantly more at all dynamicism levels when expected utility-based strategy agents were included in the simulation (except for a totally static environment, according to t-testing).

This is likely because lean agents are less likely to abandon their current jobs than greedy agents, as seen above in Chapter 3, and, since expected utility-based foremen are far more likely to successfully complete jobs, as evidenced by their greater total earnings, agents that stick with successful foremen are more likely in turn to improve their earnings.

Figures 13 – 15 show the performance of the same heuristic-based strategies from Figure 10, but with the addition of one of the following new classes: Master, Oracle, or MasterOracle. As with Figure 12, the experimental parameters laid out in Table 2 were used for the following experiments, except that rather than 2000 agents being assigned into 10 classes, the 2000 agents were assigned into 11 classes. Figure 16 shows the earnings of the expected utility-based strategy, Master and Oracle strategies as percentages of the MasterOracle strategy for comparison purposes.



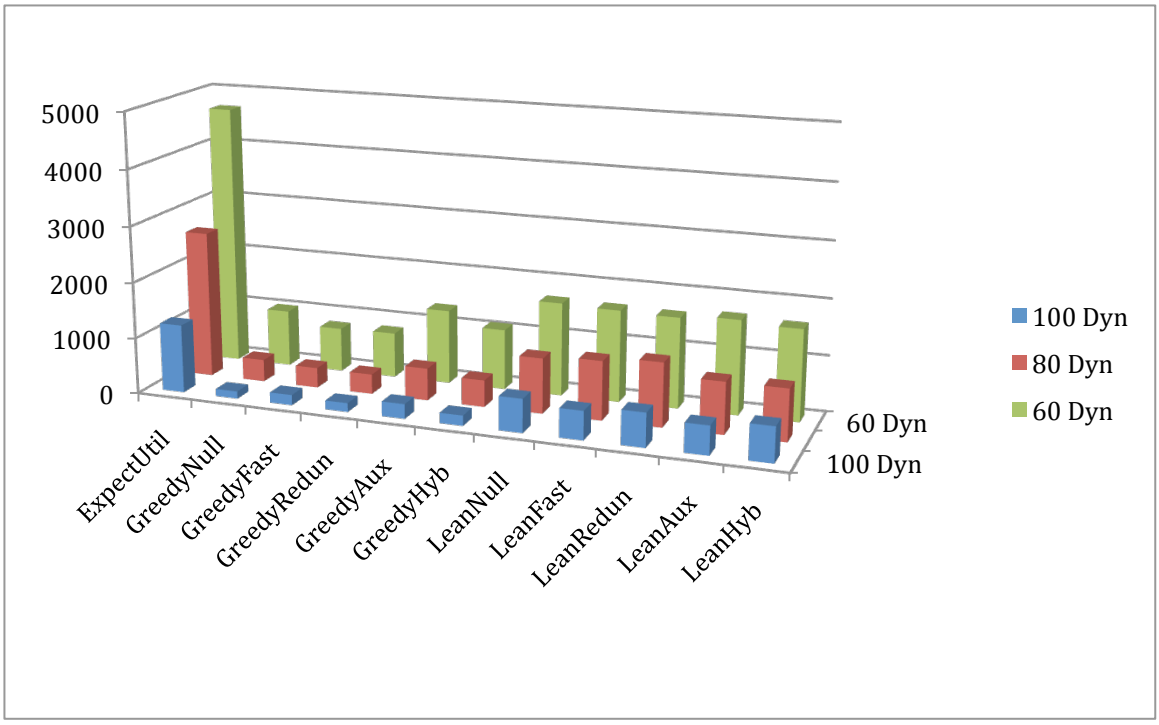
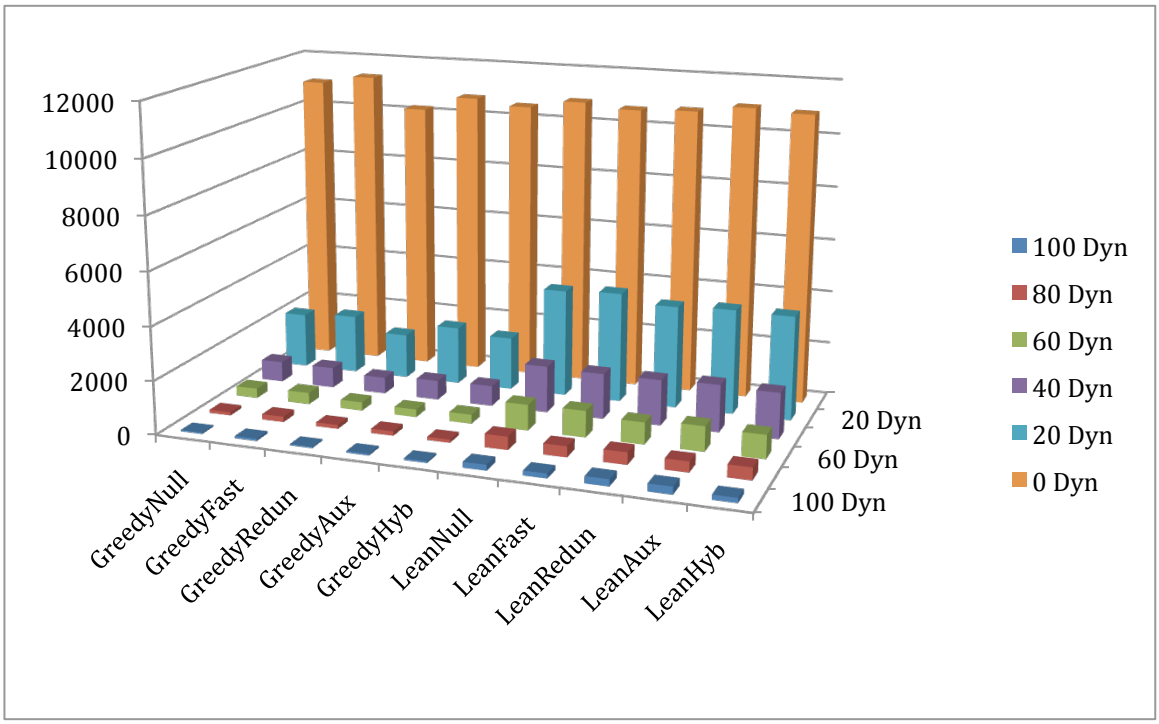


Figure 11 a-b: Average total profit by class and dynamicism level including expected utility-based strategy



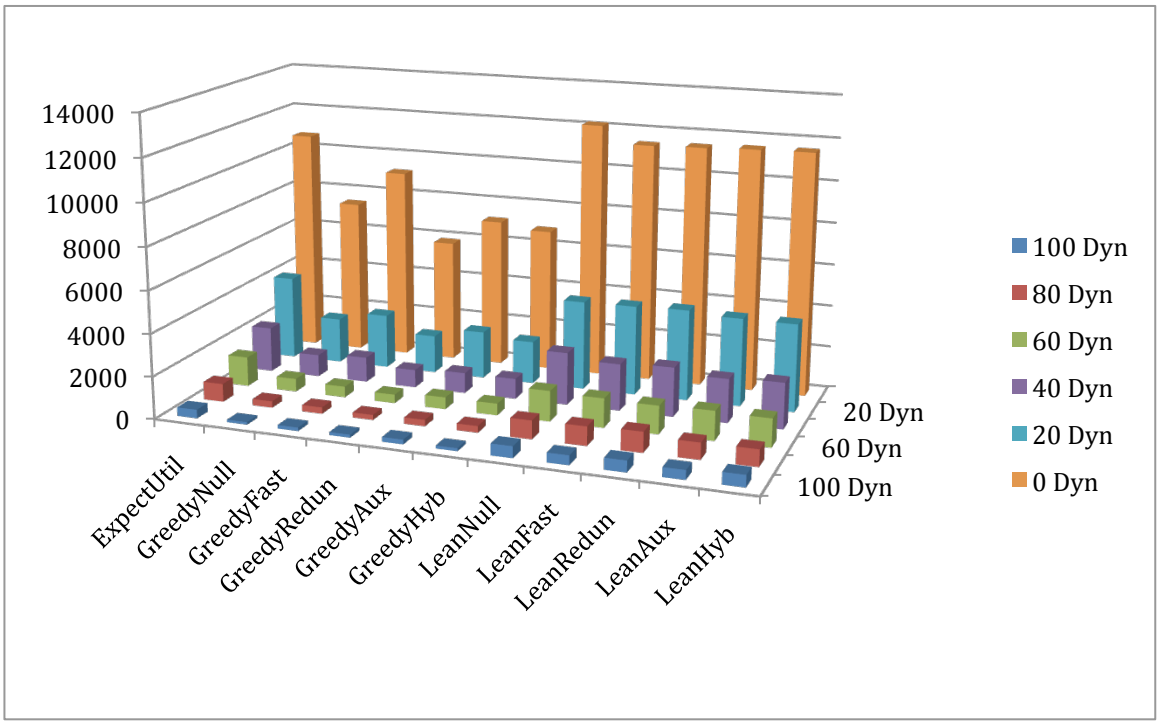
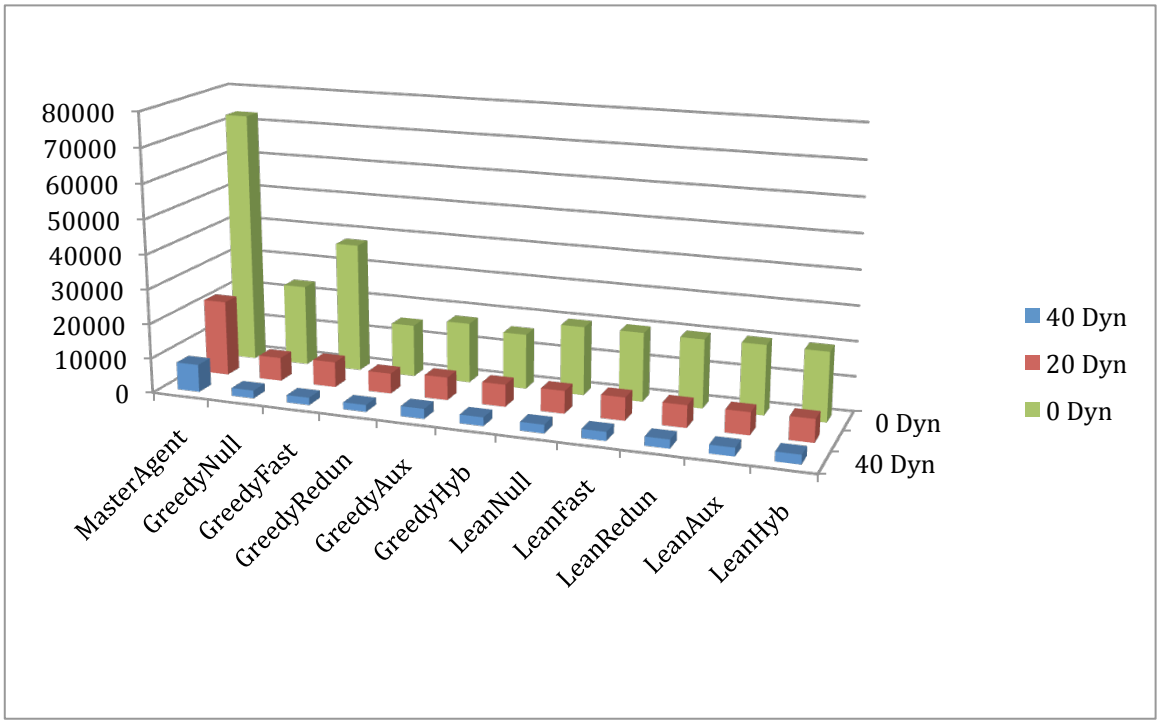


Figure 12 a-b: Average worker profit by class and dynamicism level including expected utility-based strategy



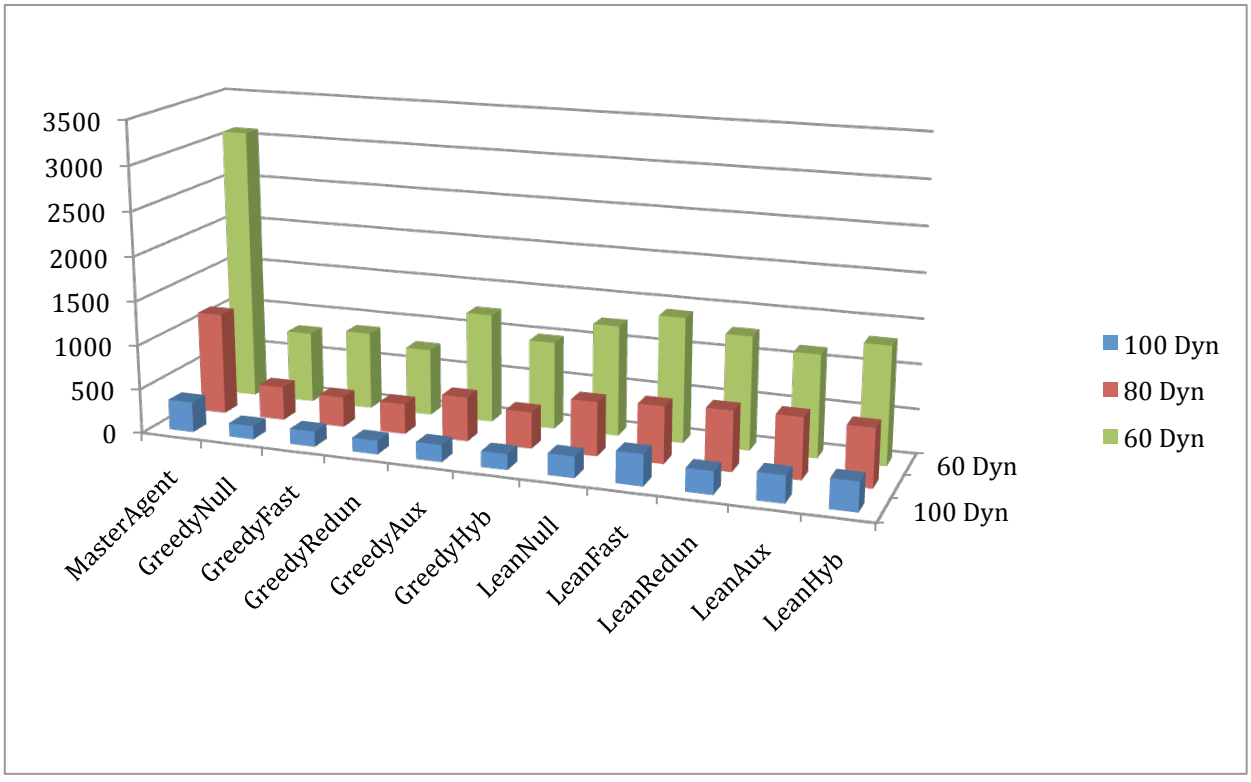
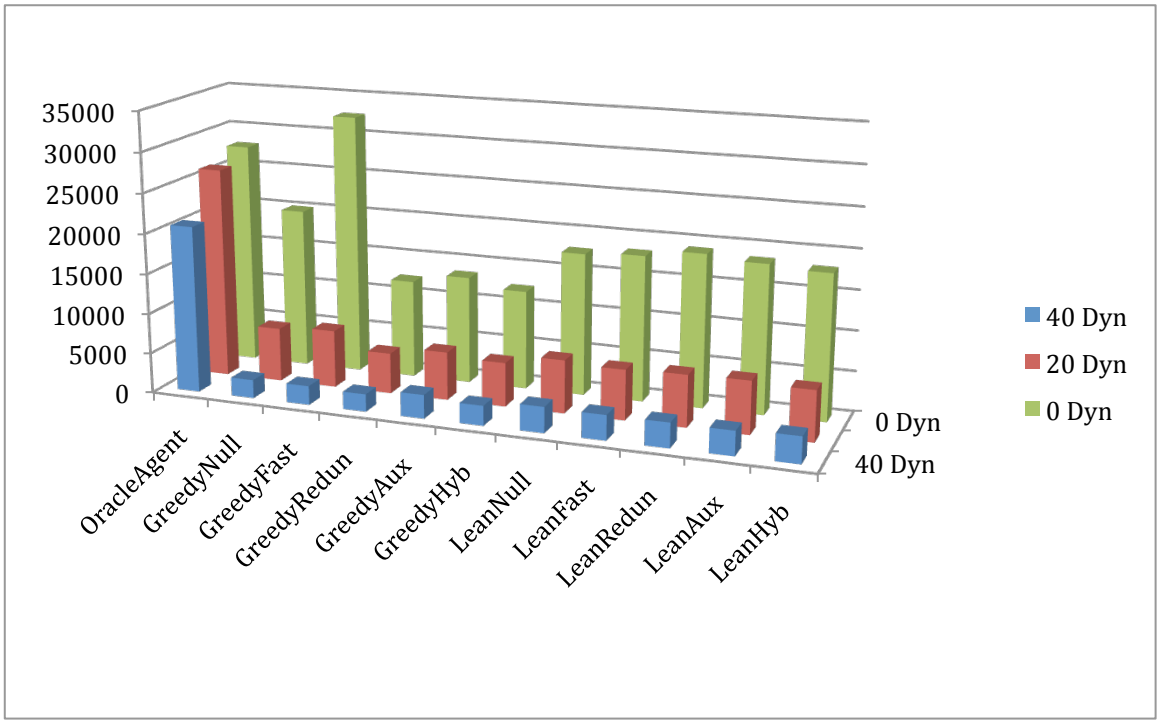


Figure 13 a-b: Average total profit by class and dynamicism level including Master strategy



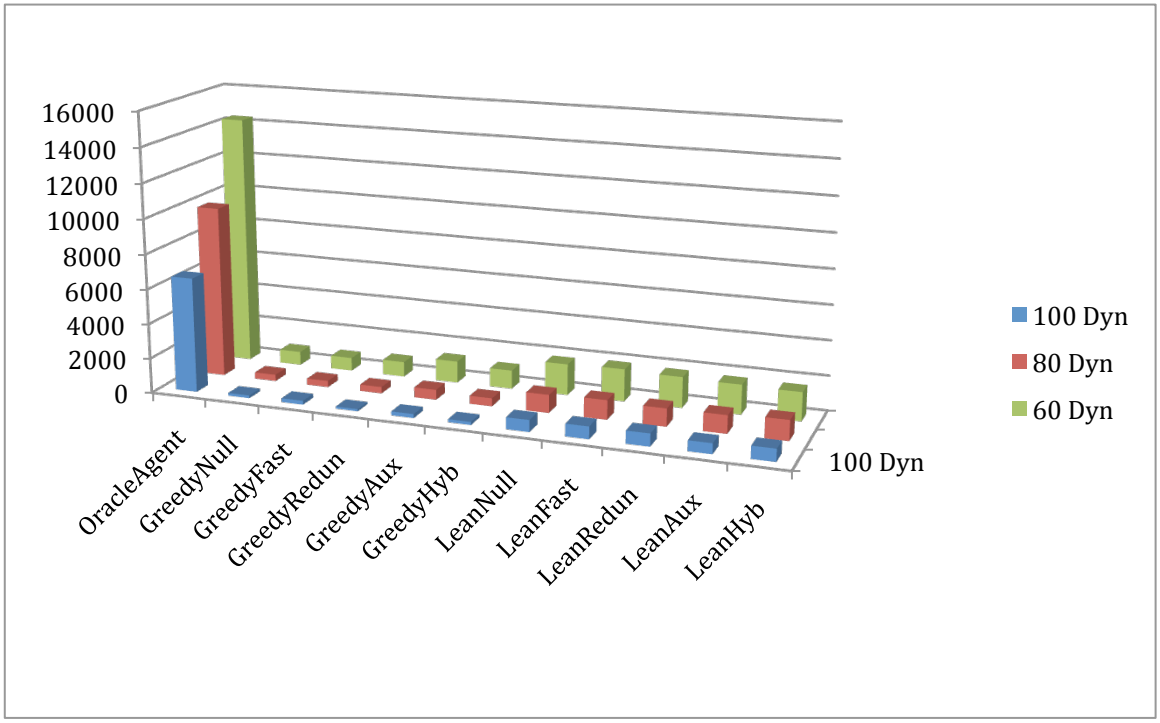
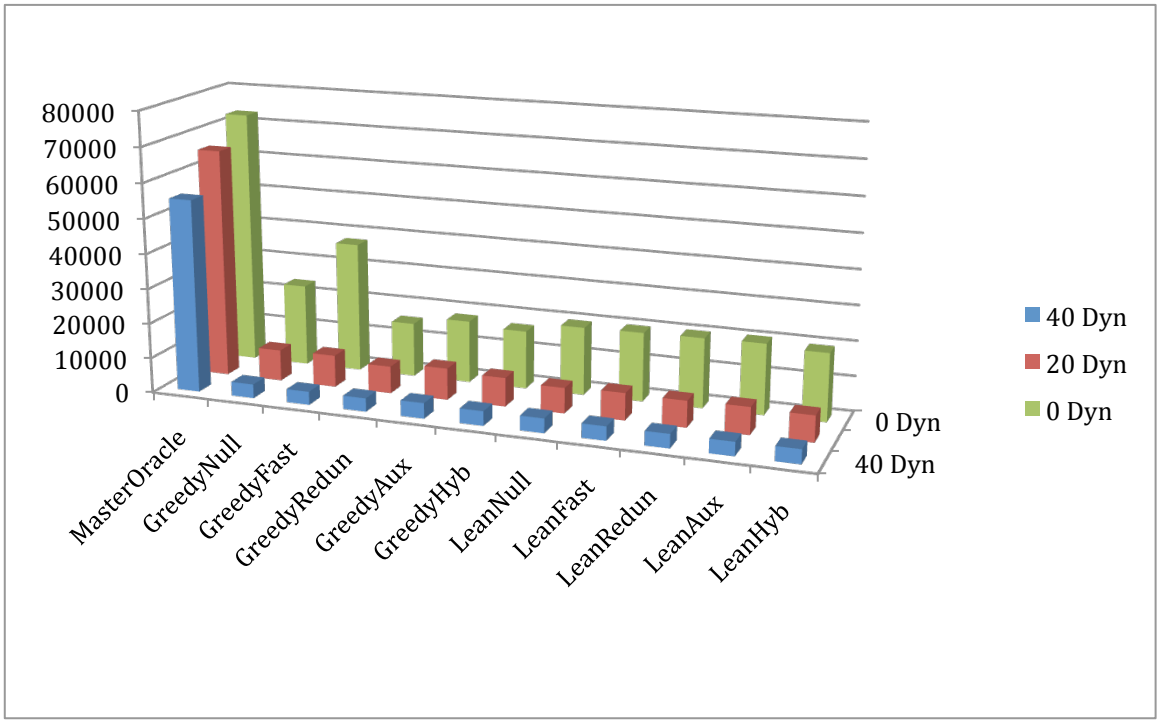


Figure 14 a-b: Average total profit by class and dynamicism level including Oracle strategy



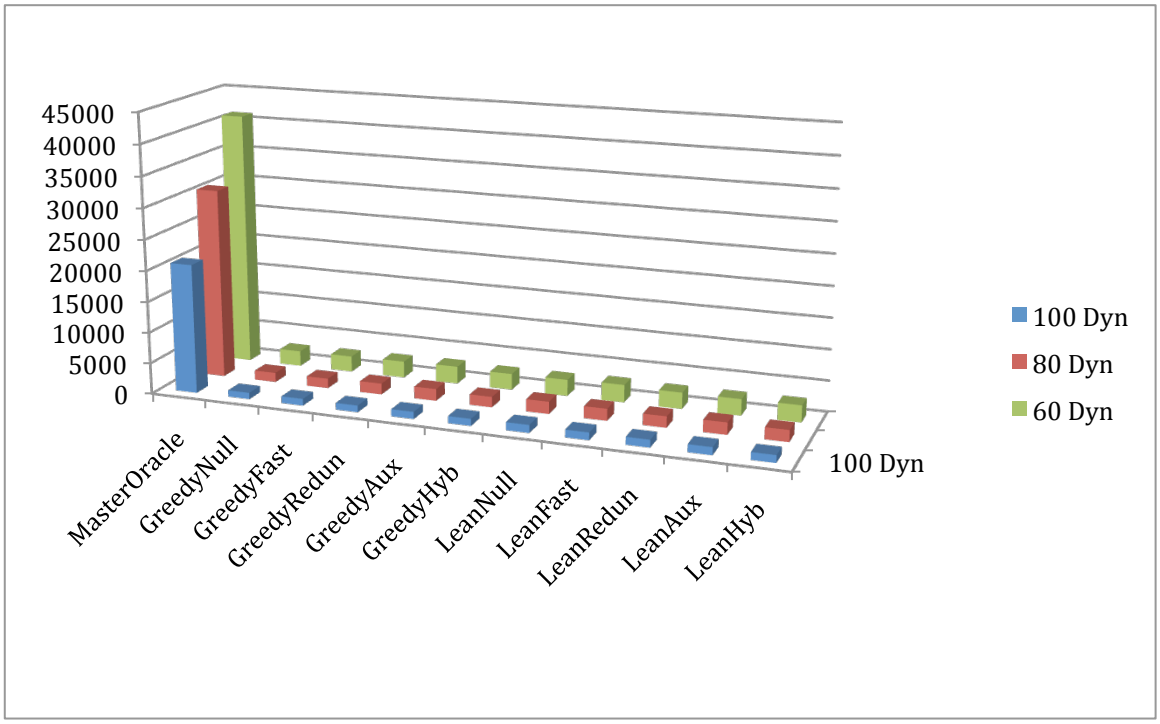


Figure 15 a-b: Average total profit by class and dynamicism level including MasterOracle strategy

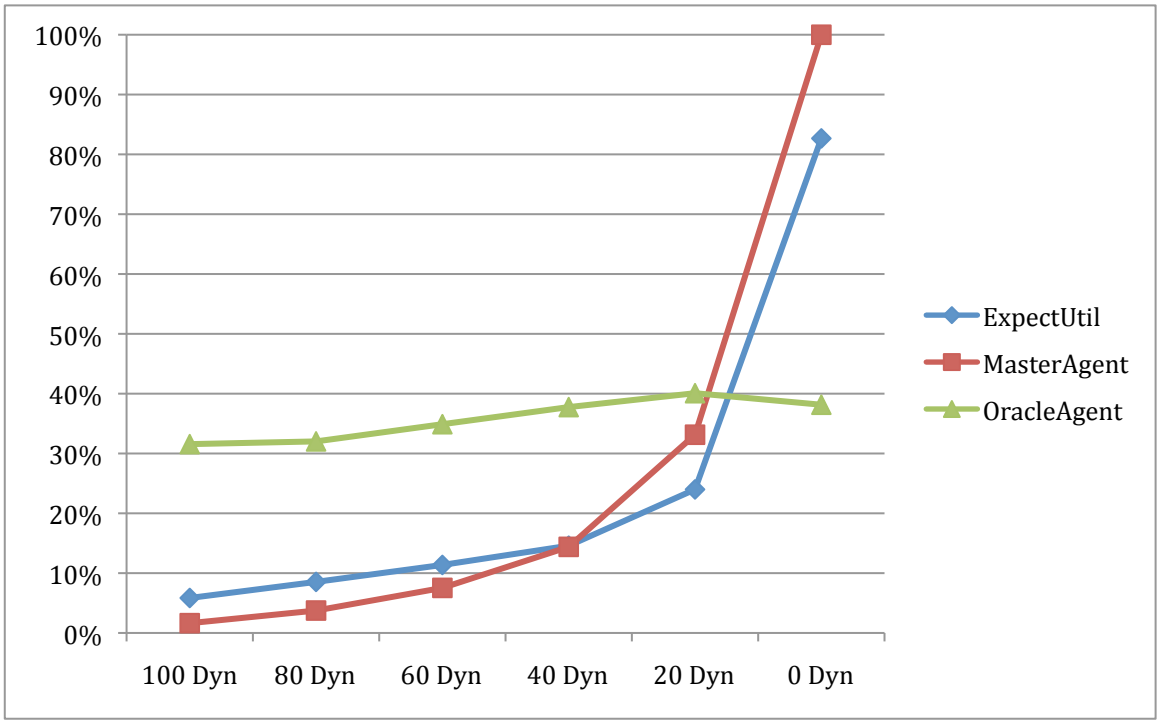


Figure 16: Strategy Earnings compared to MasterOracle Strategy

As can be seen in figures 10 – 16 above, the MasterOracle strategy unsurprisingly outperforms all other strategies when any degree of dynamicism is introduced into the environment. Moreover, the MasterOracle strategy outperforms both the individual Master and Oracle strategies in virtually all circumstances, as verified by ANOVA and t-testing[3]. This is unsurprising, since the MasterOracle agent does not deal with the same primary issues – unpredictably changing tasks and defecting partners – that the other strategies must deal with.

The Oracle strategy outperforms the Master and expected utility-based strategies at all dynamicism levels except for a purely static environment, and, in fact, maintains a constant percentage of roughly 30 to 40 percent of MasterOracle earnings at all dynamicism levels, as seen in Figure 16. Because the Oracle strategy can predict job

[3] An obvious exception is the Master strategy at 0% dynamicism, since the Master and MasterOracle strategies are effectively the same at this point, because the Master and MasterOracle strategies both have perfect knowledge about job requirements.

requrements, this is likely because a relatively high percentage of teams in the Oracle simulations fail due to defections, which is not an issue for the MasterOracle strategy.

Note, however, that the advantage the Oracle agent possesses – effectively being able to see into the future – is not likely to be found in many real-world environments. In contrast, it is possible to form structures where agents have virtually no probability of defection (although not in all domains, and not without some overhead cost.) That being the case, it is noteworthy that the performance of the expected utility-based strategy tracks the performance of the defection-free Master strategy so closely, and, at higher dynamicism levels, even surpasses it. This strongly suggests that the risk of building teams of defection-prone agents can be managed, at least to some extent, and that creating flexible teams that can deal with changes in task requirements may be even more important than creating teams where defections are not possible.

Direct comparison of the performance of the expected utility-based strategy to the MasterOracle strategy shows that the expected utility-based strategy is within 80% of the earnings of the MasterOracle strategy in a static environment, and steadily decreases from approximately 25% of MasterOracle earnings at 20% dynamicism to approximately 6% of MasterOracle earnings at 100% dynamicism. That being the case, it is noteworthy that the EU strategy, which operates in a highly uncertain, defection-prone environment, is generally within one order of magnitude as successful as the MasterOracle strategy, which faces neither of the two primary difficulties the expected utility-based strategy was designed to deal with.

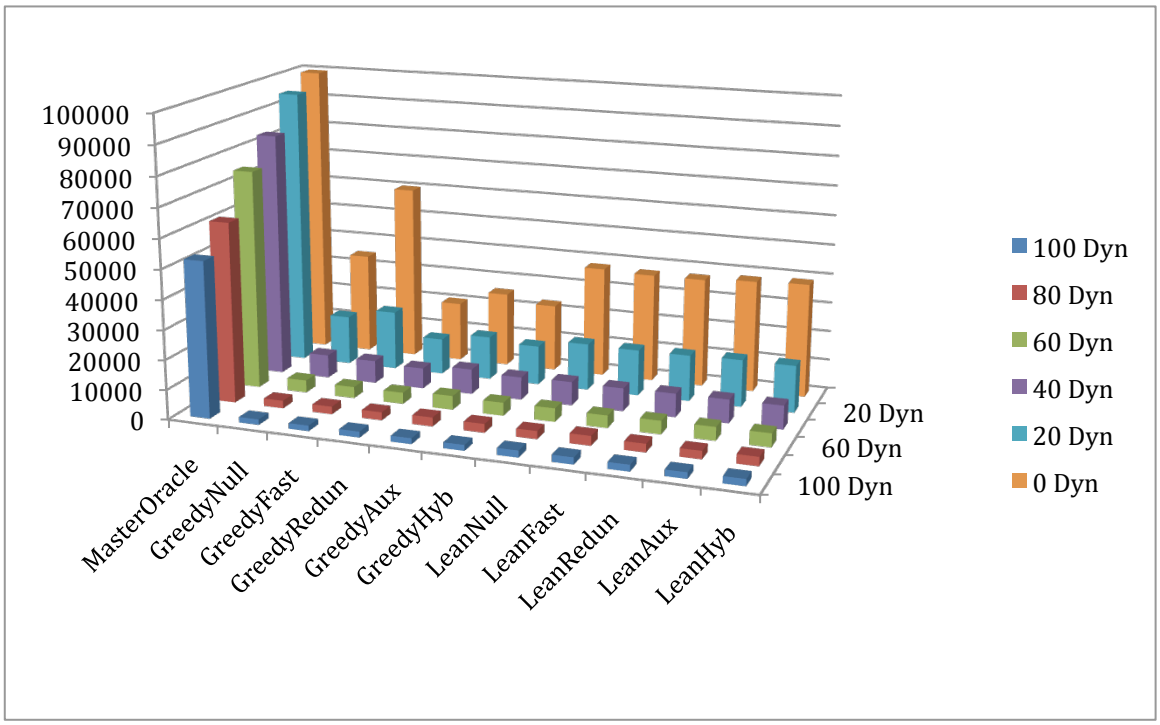
To further verify the utility and performance of the expected utility-based strategy relative to the MasterOracle strategy, a further series of experiments was carried out using the parameters described below in Table 3. More precisely, five different experiments were carried out by taking the baseline parameters from Table 2, and altering those parameters in one specific direction to explore different types of environments and settings: environments where jobs begin with a greater number of required tasks, environments where the number of possible tasks and skills is much greater, environments where agents are scarce compared to the number of jobs, environments

where jobs are scarce compared to the number of agents, and environments where agents have fewer skills, compared to the total number of possible skills.

Table 3: Additional Experimental parameters for Figures 17 - 22

	Baseline	Many Agent Skills	Many Global Skills	Scarce Agents	Scarce Agent Skills	Scarce Jobs
Total Agents	2000	2000	2000	200	2000	2000
Total Jobs	1000	1000	1000	1000	1000	100
Global skills	10	10	20	10	10	10
Skills per agent	5	8	5	5	2	5

Figures 17 – 21 provide the results of experiments carried out using the various parameters listed in Table 3. Specifically, Figure 17a shows the ten heuristic strategies operating alongside the MasterOracle strategy using the Many Agent Skills parameters from Table 3, while Figure 17b shows the expected utility-based strategy executing alongside the heuristic strategies in an experiment using the Many Agent Skills parameters. Similarly, Figure 18 shows a comparison of the MasterOracle and expected utility-based strategy using the Many Global Skills parameters, Figure 19 shows a comparison of the strategies with the Scarce Agents parameters, Figure 20 a comparison with the Scarce Agent Skills parameters, and Figure 21 a comparison with the Scarce Jobs parameters. Finally, Figure 22 shows the expected utility-based strategy earnings as a percentage of MasterOracle strategy earnings for all five experimental parameter sets in Table 3.



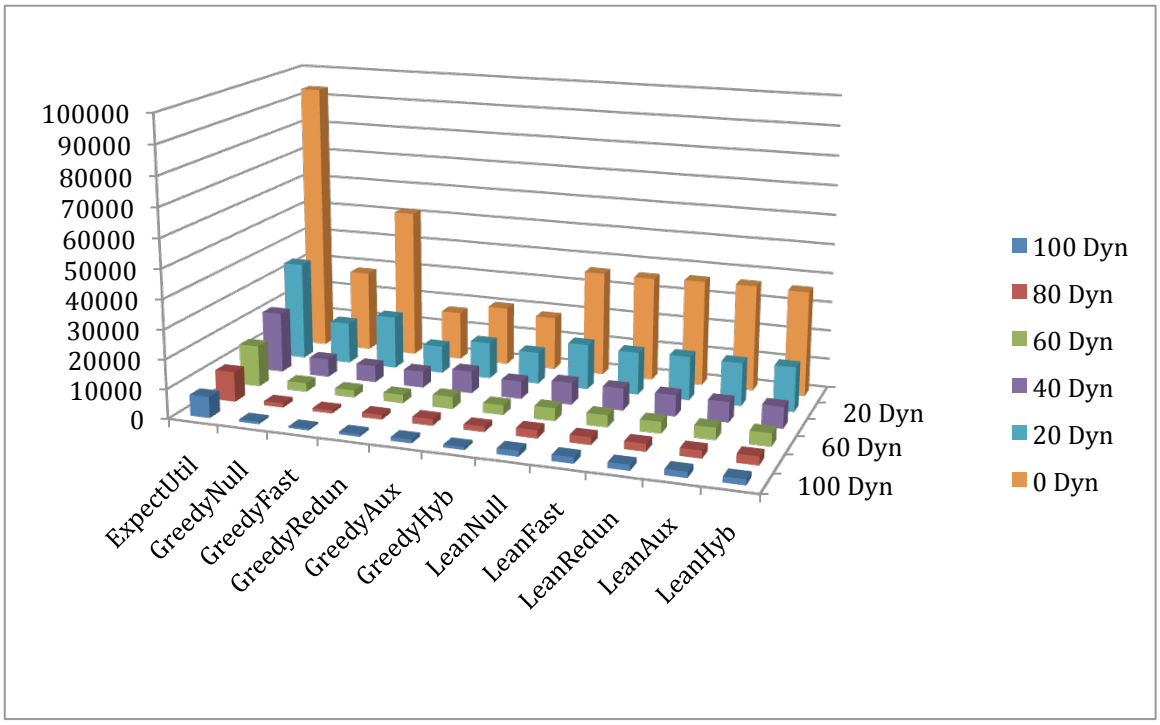


Figure 17 a-b: Master Oracle and expected utility-based strategies with Many Agent Skills parameters

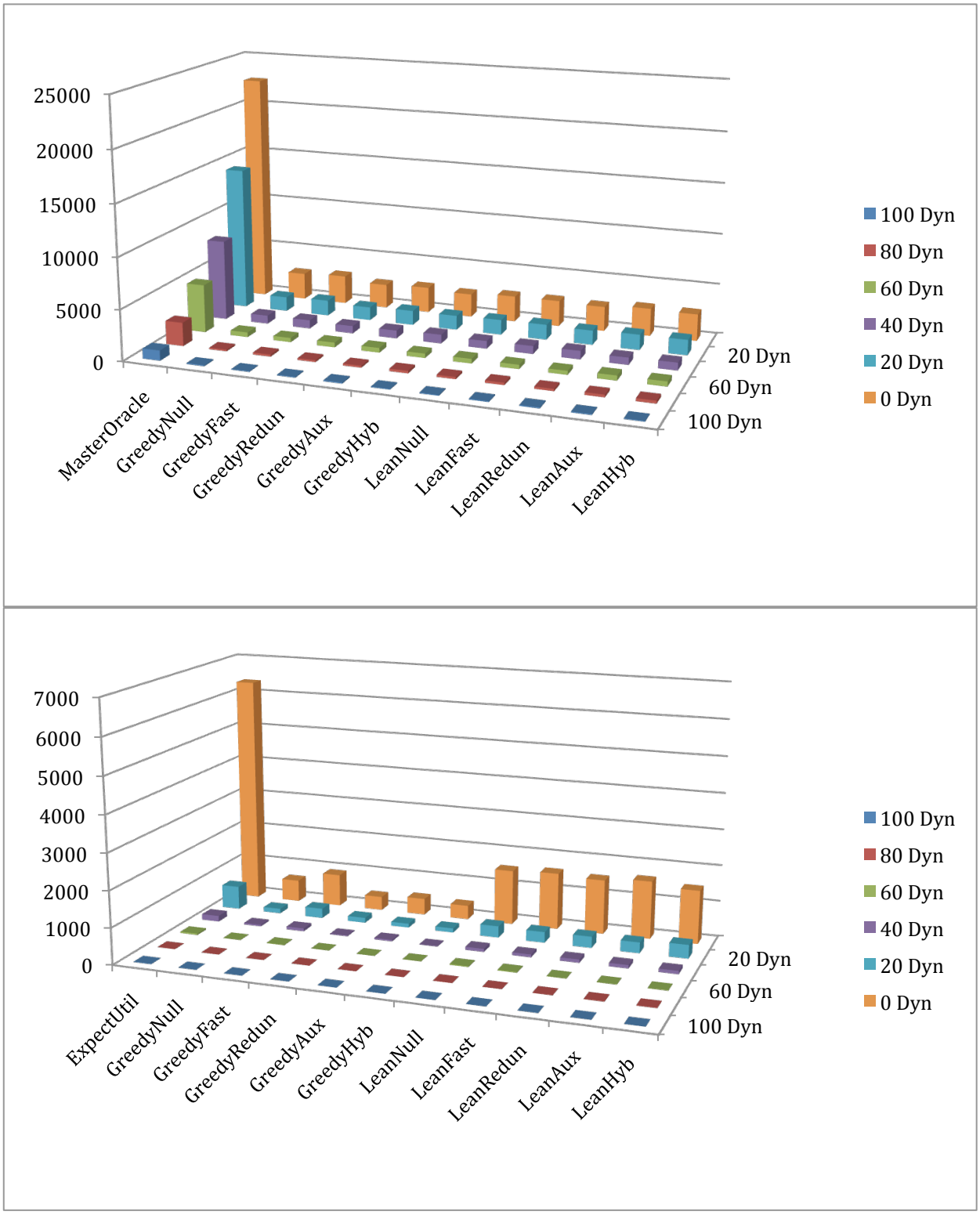


Figure 18 a-b: Master Oracle and expected utility-based strategies with Many Global Skills parameters

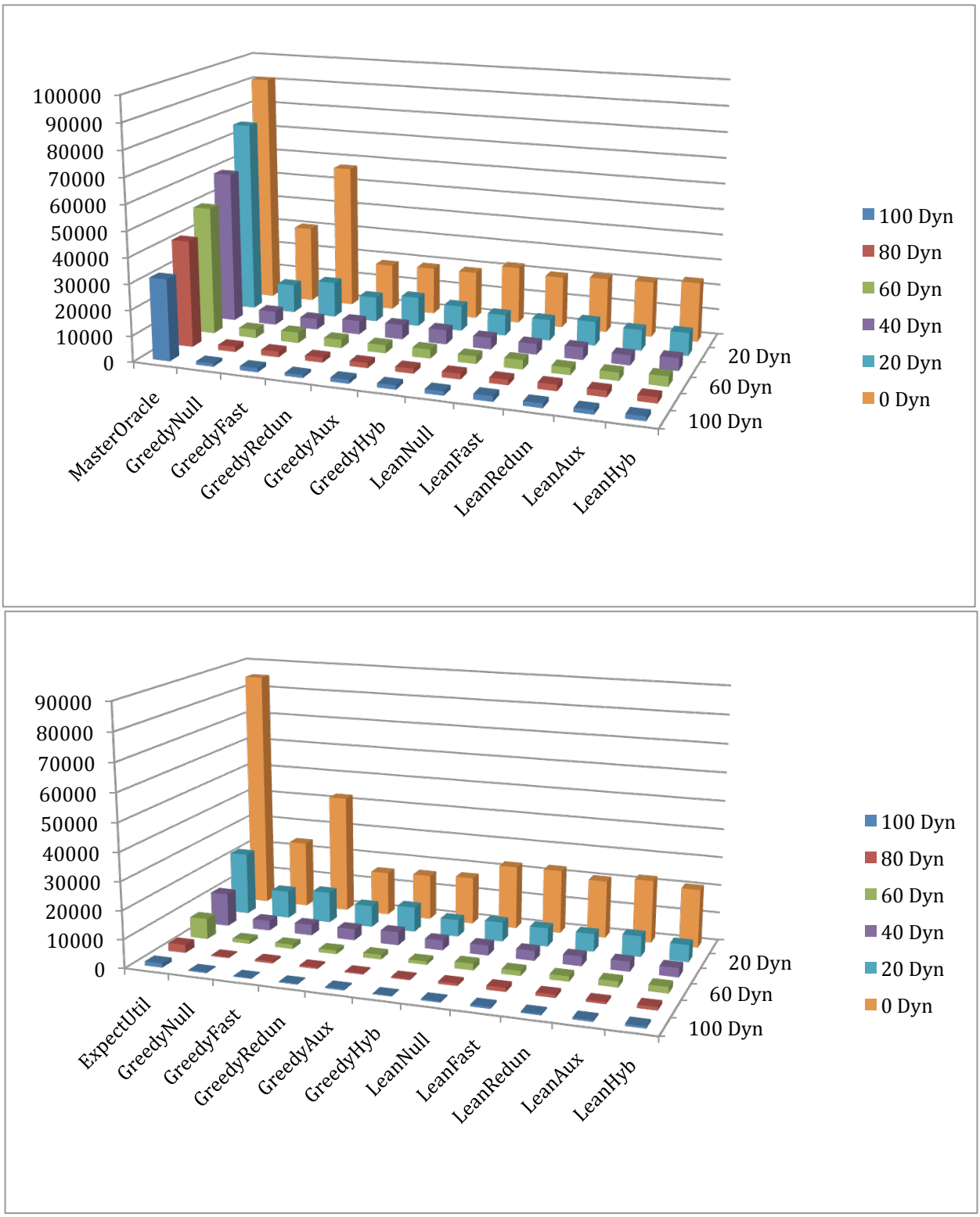


Figure 19 a-b: Master Oracle and expected utility-based strategies with Scarce Agent parameters

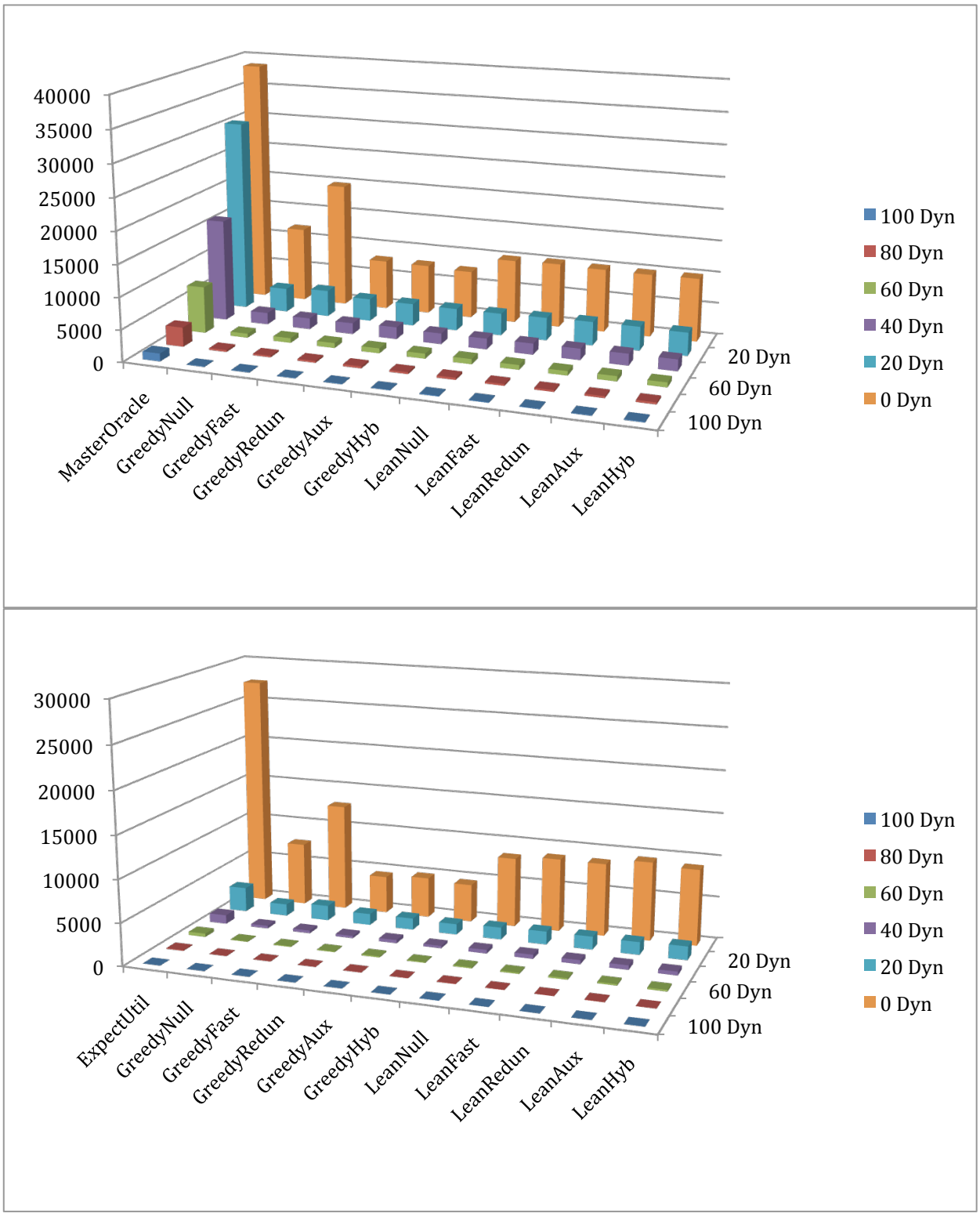


Figure 20 a-b: Master Oracle and expected utility-based strategies with Scarce Agent Skills parameters

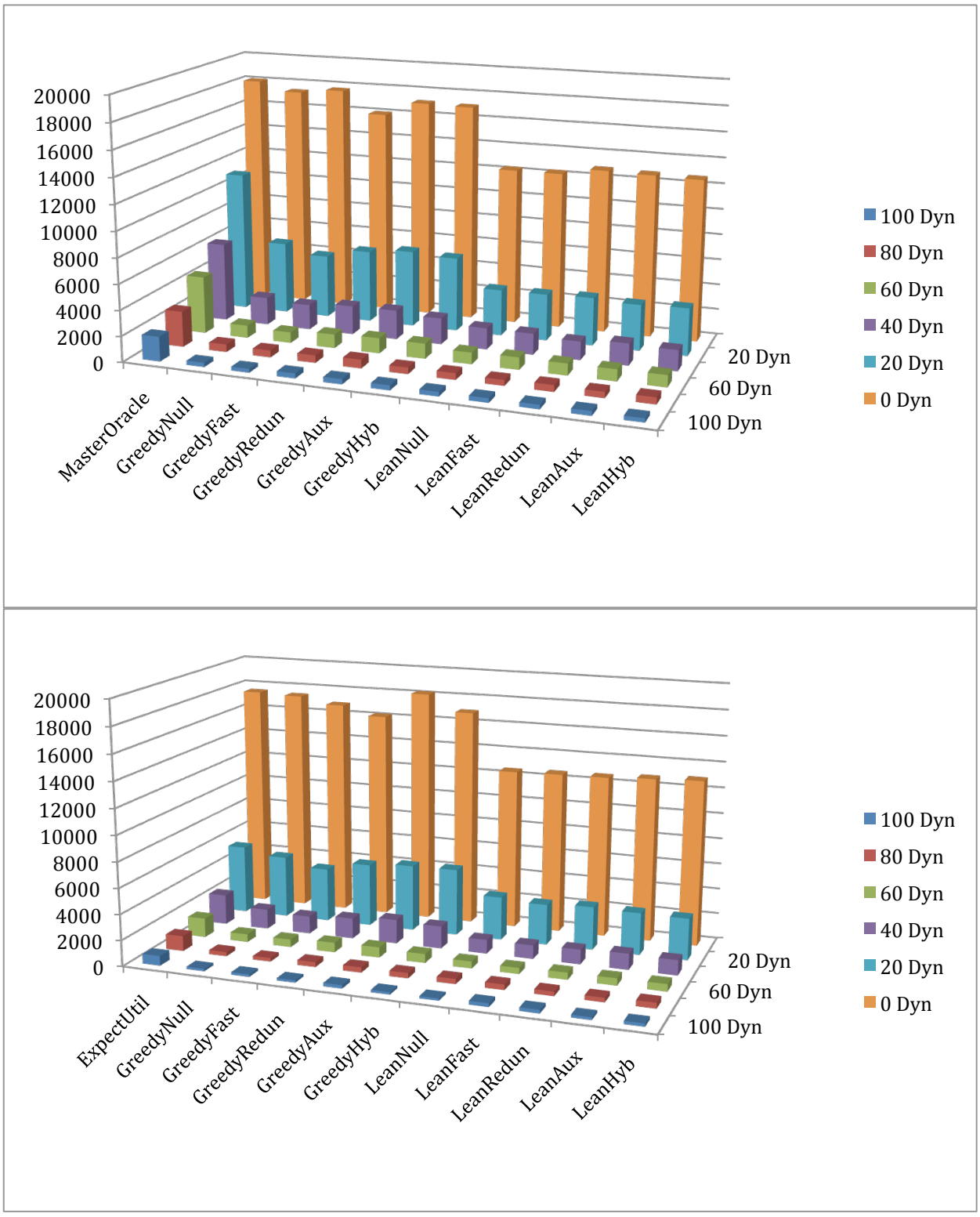


Figure 21 a-b: Master Oracle and expected utility-based strategies with Scarce Jobs parameters

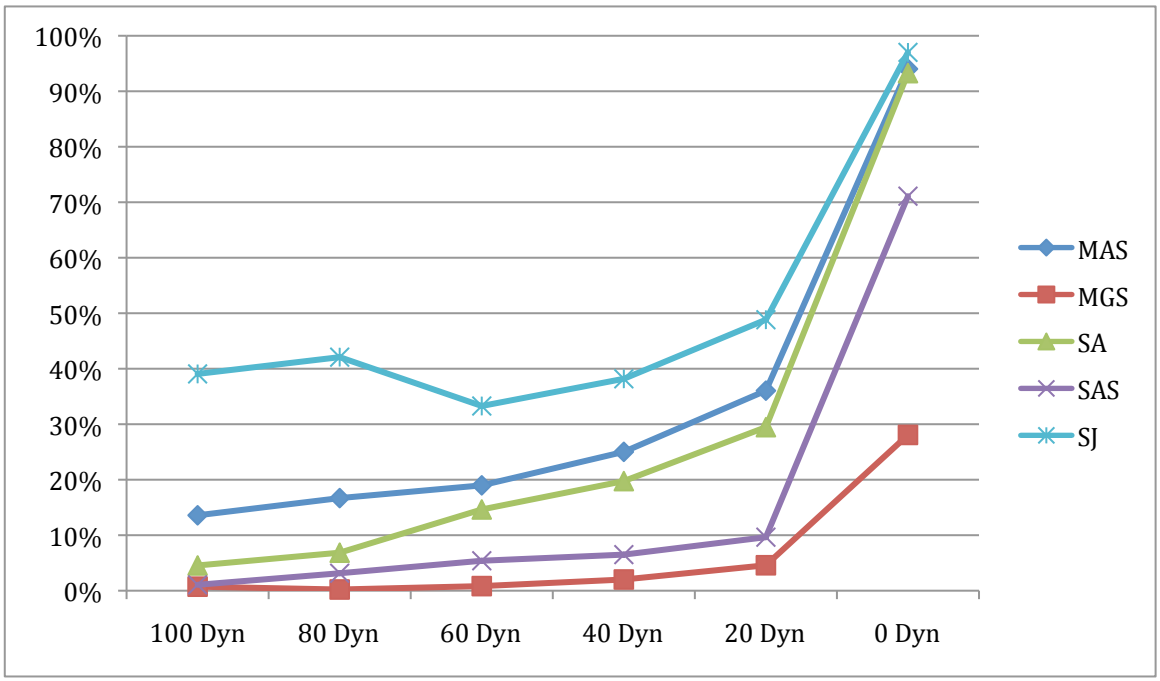


Figure 22: Expected utility-based strategy compared to MasterOracle for alternate experimental parameters

As can be seen from Figures 17 – 22 above, the broad pattern seen in the baseline experiments continues to hold true for the alternate experimental parameters found in Table 3: while the expected utility-based strategy generally outperforms the preexisting heuristic based strategies, the MasterOracle strategy dominates the EU strategy in earnings. Pairwise t-testing of MasterOracle and EU strategy earnings for all alternate test parameters shows that MO significantly outperforms EU on all parameters and at all dynamicism levels, except for the MAS, SA, and SJ parameters when D = 0%.

Of somewhat greater interest is how the EU strategy functions in different types of experiments. ANOVA analysis indicates a statistically significant spread between all heuristic strategies and the EU strategies at all parameters and dynamicism levels, except for the Many Skills parameters at 80% and 100%, as will be discussed in further detail below. The EU strategy comes closest to competing against the MasterOracle strategy in the Scarce Jobs experiments – in fact, in completely static environments, both the MasterOracle and EU strategies are nearly on par with the regular heuristic strategies, and

in low dynamicism environments, the EU strategy is only slightly better than the heuristic strategies. This is likely because, in an environment where competition for jobs is fierce, agents have less of an opportunity to defect from their current job, making it easier for any foreman agent, regardless of its strategy, to succeed.

The Scarce Agent and Many Agent Skill experimental parameters are broadly similar in result to the baseline parameters, both in how the EU strategy compares to the heuristic strategies, and in how the EU strategy compares to the MasterOracle strategy. However, the results of the Scarce Agent Skills and Many Global Skills experiments are interesting. Note that even the MasterOracle strategy did not perform well in these environments, and that the MasterOracle strategy's earnings decreased rapidly as the dynamicism level was increased. Likewise, the EU strategy barely earned a profit when dynamicism was above 20%.

It is highly likely that the reason for this relative lack of performance is because of the way teams are assembled by the agent. Remember that agent all select desirable teams, according to their implemented strategy, from a pool of randomly created teams. Each randomly created team is capable of solving the current state of the job (i.e. the current active tasks with their current quality of service demands), but other factors – how adaptable the team is, how redundant the skills on the team are, how many tasks each agent is assigned – are set purely by chance, rather than crafted specifically by the agent strategies. This mechanism has the advantages of being an anytime algorithm capable of searching very large state spaces (the more potential teams an agent examines, the better the team it can ultimately select), and of making it easy to compare different strategies (since strategies ultimately only control what jobs and teams the agent selects, and all other aspects of agent behavior are the same).

However, in environments where the skills the agents possess are few in number relative to the total number of global task types (which holds true for both Scarce Agent Skills and Many Global Skills), the chances that a randomly assembled team will be able to cover a significant portion of the tasks that may become active is significantly decreased, since the random, unused skills that are crucial to team adaptability, but are completely ignored by the team generation mechanism, are largely absent. Accordingly,

it seems reasonable to suggest that future work in this area should explore the explicit creation of teams by an agent strategy to ensure that excess skills which allow for team adaptability (among other beneficial team properties) will be present.

5.2 Accuracy of Expected Utility-based strategy

Previous experiments have established that the expected utility-based strategy performs well relative to existing heuristic-based strategies, and generally earns within an order of magnitude compared to the optimal MasterOracle strategy. However, beyond the metric of agent earnings, how well do the individual components of the expected utility-based strategy perform? More particularly, the expected utility-based strategy is composed of three high-level estimates about a job/team combination: what are the expected earnings from the job for the EU agent assuming the job is completed, what is the likelihood that the job will be successfully completed, and what is the expected duration of the job, assuming the job is completed?

To investigate the accuracy of these estimates, we created instrumentation in the expected utility-based strategy that would keep track of the initial estimates agents made about job/team combinations, and the ultimate values of the estimated quantities. Specifically, after an EU agent selected a job to work on, the EU agent would store the expected earnings value, probability of success, and expected duration of the job. Upon successfully completing the job, the agent would compare the ultimate earnings and duration of the job to the estimate, and store the difference between the two. By averaging the difference between the estimate and final value across all completed jobs and across all EU agents, we can determine the overall accuracy of these two estimations. Likewise, by comparing the average of the estimates of job success to an EU agent's actual success in completing all attempted jobs, we can determine the overall accuracy of the EU strategy's probability of success estimate.

Accordingly, Figures 23, 24, and 25 show the average difference between the estimate and actual values of job earnings, job duration, and probability of success, respectively, as taken from the same experimental trials whose parameters where laid out

in Table 2, and whose results were shown in Figure 11. Note that because these Figures show the difference between the estimated value minus the actual value, a positive value in the figure means the estimated value was greater than the final value, and a negative value means the estimated value was less than the final value.

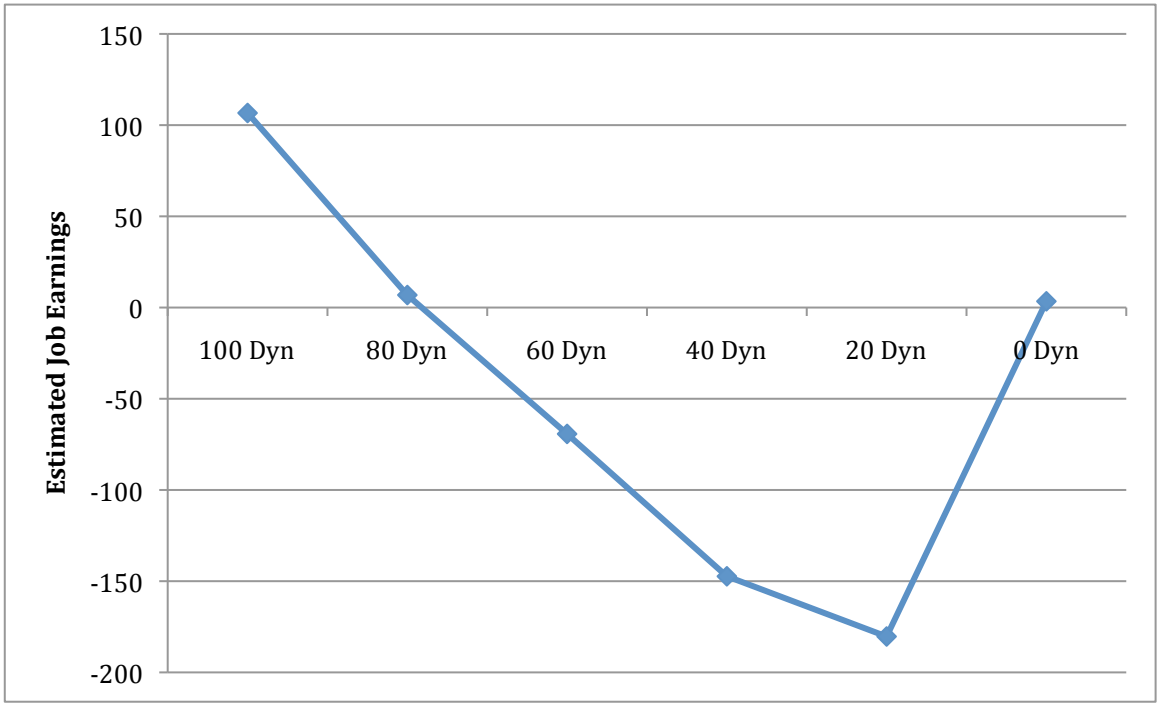


Figure 23: Average difference in expected utility strategy's estimated job earnings and actual job earnings

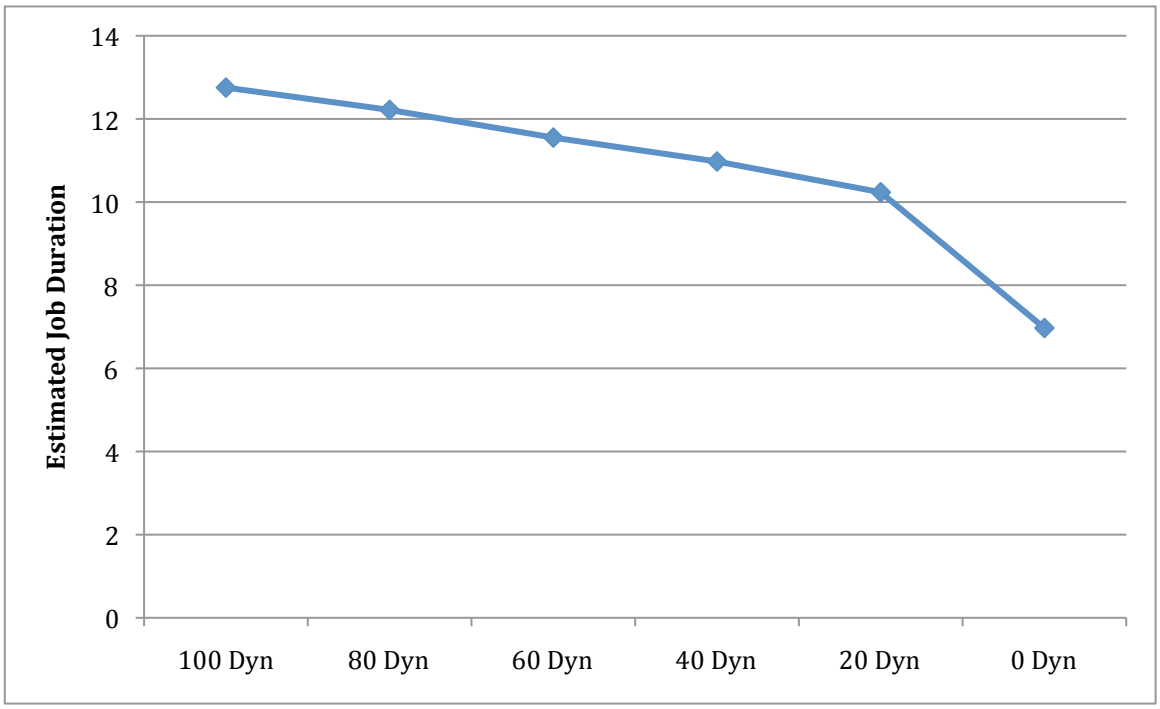


Figure 24: Average difference in expected utility strategy's estimated job duration and actual job duration

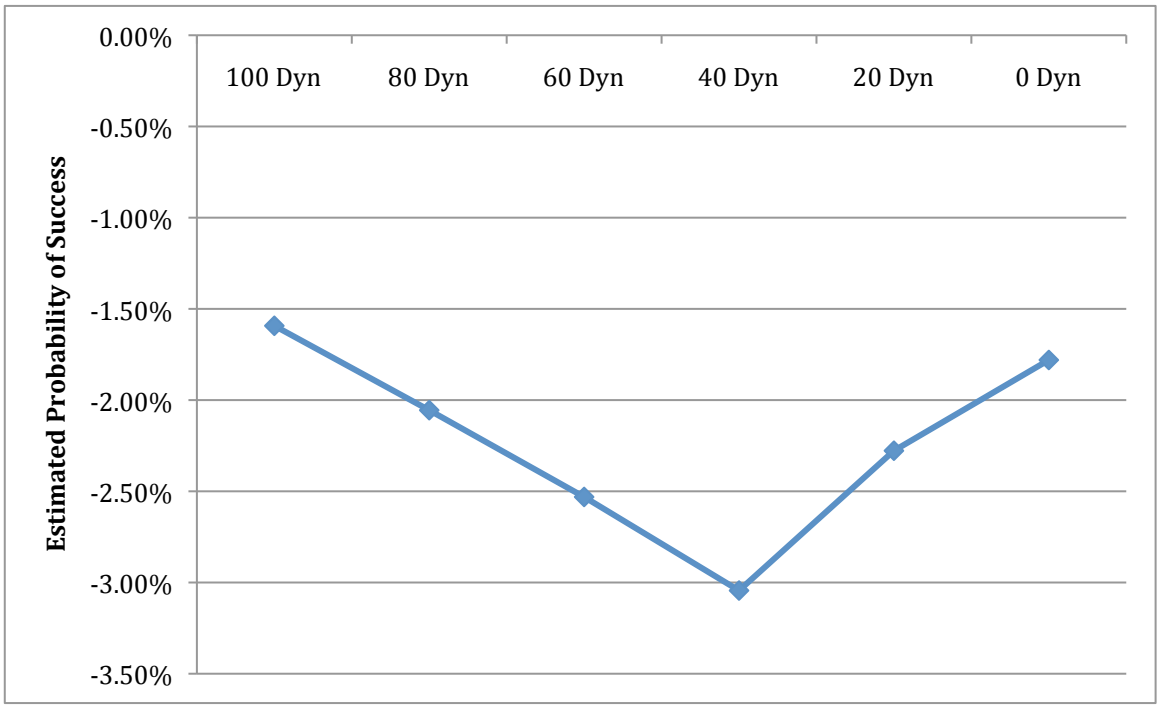


Figure 25: Average difference in expected utility strategy's estimated probability of success and actual percentage of attempted jobs successfully completed

The results shown in Figure 23 show that the expected utility-based strategy's estimate of job earnings changes somewhat with the dynamicism of the environment, but is roughly within plus or minus 150 credits of the actual value. This would seem to be a fairly accurate estimate, given that the average value of a task is approximately 1220 credits – meaning that, at worst, the average inaccuracy of the EU strategy's earnings estimate is slightly more than 10% of the value of a single task.

The results shown in Figure 24 indicate that the expected utility-based strategy consistently overestimate the duration of an attempted job by roughly 12 timesteps per job (although this figure is lower for the 100% dynamicism level) – given that the average length of a task is 6 time units, this means that the EU strategy initially overestimates the duration of a task by two entire tasks. It is noted that some degree of overestimation might be expected from a methodology borrowed from the project management domain, where the costs for delivering a project late far outweigh the costs of delivering a project early, leading to a strong preference for overly conservative estimates. Furthermore, the initial duration estimate is essentially an outlier for job duration, since as the job progresses and more tasks become completed or permanently inactive, the duration estimate improves and the final estimate approaches the actual duration of the job. However, it is likely that further experimentation in this area could improve the accuracy of job duration estimations.

The results shown in Figure 25 show a consistent underestimation of the probability that a job will be completed by the EU strategy, with the difference in estimation and actuall success varying between 1.5% and 3% over all dynamicism values. From an absolute perspective, even a 3% inaccuracy in the probability of completing a job seems relatively minor. In addition, it is likely better to be overly pessimistic about the chances of completing a job than overly optimistic, since such pessimism will likely keep agents from quitting viable teams in order to pursue all but the most outstanding and sure-fire opportunities.

Furthermore, this "V" shape is understandable, given that a largely static environment is easier to predict, as is a 100% dynamic environment, since it is known (or highly suspected) that *no* task instance will actually yield it's initially apparent payout. In contrast, the 40% and 60% dynamicism ranges are more difficult to predict, since the chances of a task instance changing or not are nearly even.

5.3 Performance of Learning Mechanisms

Experimental results presented in previous sections have established that the expected utility-based strategy outperforms existing heuristic-based strategies in many environments, and is usually within an order of magnitude of strategies such as the MasterOracle strategy, which operates with advantages such as advance knowledge of task changes and perfectly reliable partner agents.

However, in these experiments, the EU strategy operated with the advantage of knowing the precise dynamicism value, D, of the environment it was embedded in, as well as the general reliability of other agents. In real world domains, although it may be possible to characterize how likely job requirements are to change and/or how generally reliable other agents are, it is possible that such data would not be available to an agent.

Therefore, in this section, the research assumes that such data is not a priori available. Accordingly, agents executing the EU strategy require learning mechanisms to determine values such as D and agent reliability from direct observations of the environment. Section 4.3 provided the theory on how agents learn such environmental variables, specifically showing how the expected utility function could determine the value of D for a single observed job, and further describing mechanisms for how such observations could be weighted to better determine a changing value of D.

This section therefore presents the results of experiments designed to show how effective these learning mechanisms are, both in terms of how they compare in earnings to an expected utility-based strategy with a provided value of D (as opposed to a learned value of D) and in how closely they track the actual value of D as it changes in the environment. Note that the results of this section focus entirely on how effective the

mechanisms for learning the value of D are, since general trust and reputation mechanisms for learning the trustworthiness and reliability of other agents are already well known (Gujral, DeAngelis et al. 2006; Fullam 2007).

We begin by comparing the earnings performance of four different EU-derived strategies operating in the environment described by Table 2 above. Specifically, the standard EU strategy was executed and informed of the set value of D in the environment. In contrast, three other versions of the EU strategy were also executed and expected to learn the value of D, rather than being directly informed. The **Running Average** strategy kept a straightforward average of all observations, while the **Sliding Window** strategy only averaged those observations of D that had occurred within the past 100 timesteps, and the **Exponential Weighting** strategy combined each most recent observation of D with the existing average of D depending on how long ago the last observation of D was taken, as shown below in Equation 29:

(Eqn. 29): $$DEstimate_{t=x} = (DEstimate_{t=x-y}(.99^{y}) + (DObservation_{t=x}(1-.99^{y}))$$

Figure 26 shows the average earnings of all four classes across a range of values of D (the static value, D = 0, was not included), while Figure 27 shows the earnings of the three new strategies as a percentage of the Known Dynamicism version of the EU strategy – that is, the version of the strategy which was provided with the value of D in advance.

As can be seen from the results below, the earnings for all three learning strategies are broadly comparable to the Known Dynamicism strategy, with the Running Average and Exponential Weighting strategies being closest to the Known Dynamicism strategy, and the Sliding Window strategy being slightly less accurate, for all dynamicism levels. More particularly, ANOVA analysis of the data shows that there is no statistically significant difference in the earnings of all four strategies at 20% dynamicism, and individual t-tests show that there is no significant difference between the Exponential Weighting strategy and the Known Dynamicism strategy at any dynamicism level but 40%, and no

significant difference between the Running Average strategy and the Known Dynamicism strategy at the 60% and 80% levels.

Figures 28 through 32 allow us to examine these results more closely by showing how precisely the three learning strategies tracked the actual dynamicism value (or "control" value) over time. More specifically, the "a" part of each figure shows the average estimated value of D for all three learning strategies across the lifetime of the experimental trials, while the "b" part of each figure shows the average difference between the estimate and the control value.

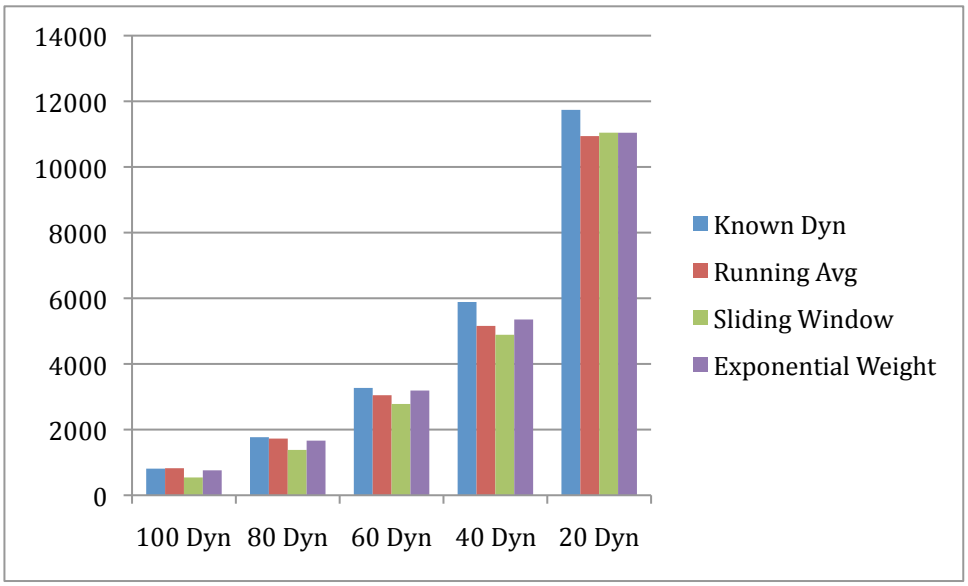


Figure 26: Average total earnings for EU learning strategies and EU with known dynamicism

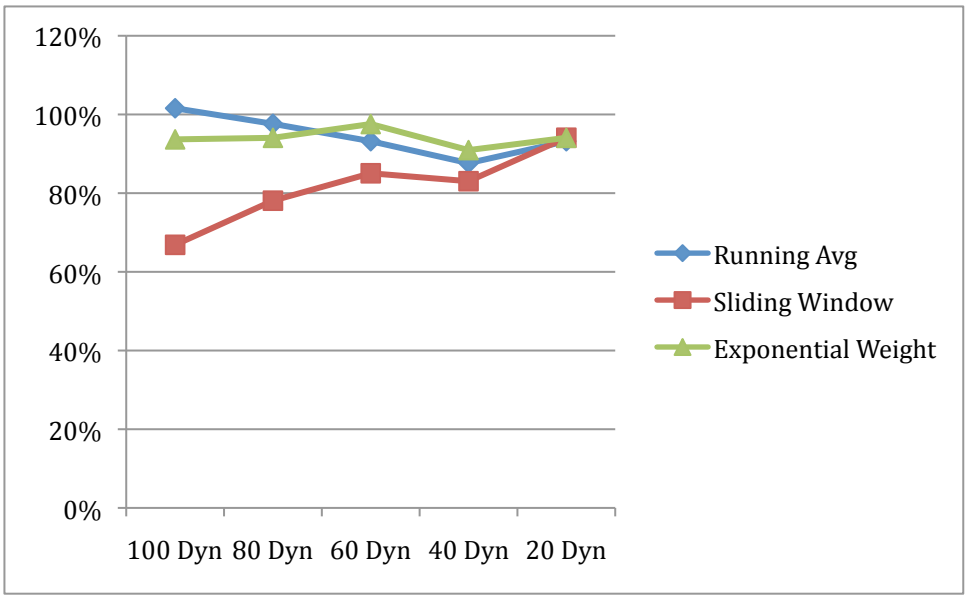
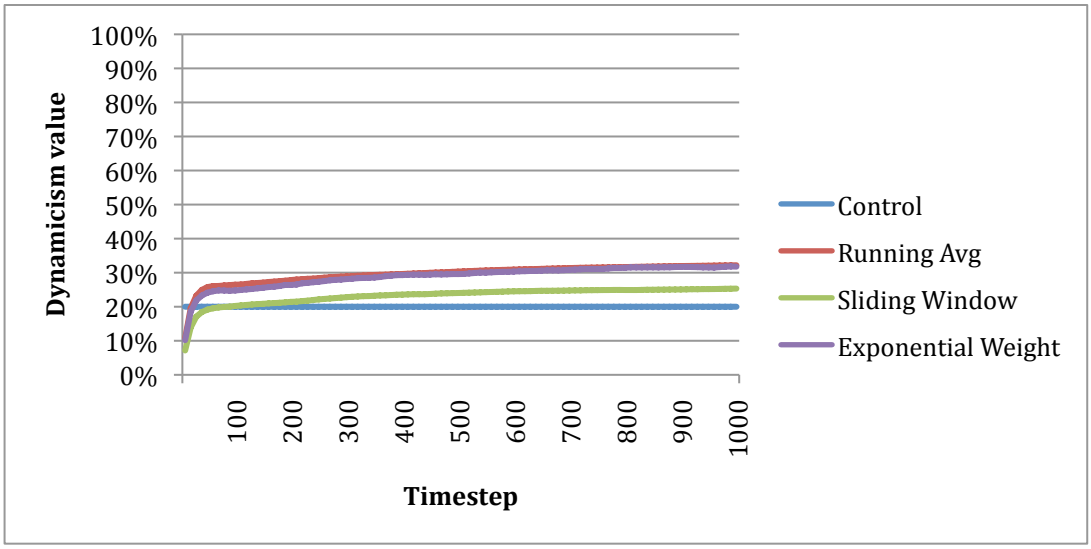


Figure 27: Average total earnings for EU learning strategies as a percentage of EU with known dynamicism earnings



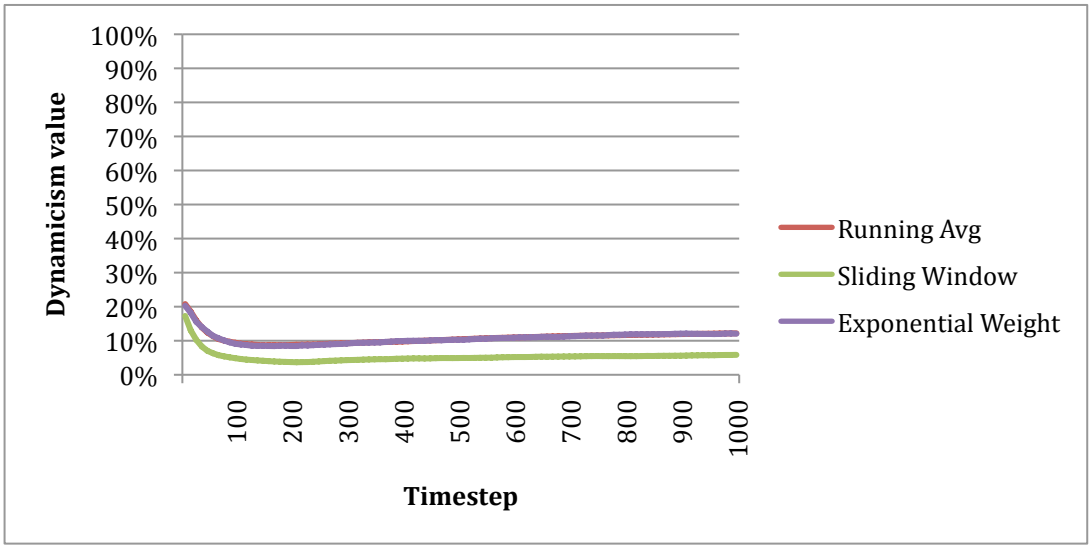


Figure 28a-b: EU learning strategies average estimation of 20% control dynamicism level and average estimation error

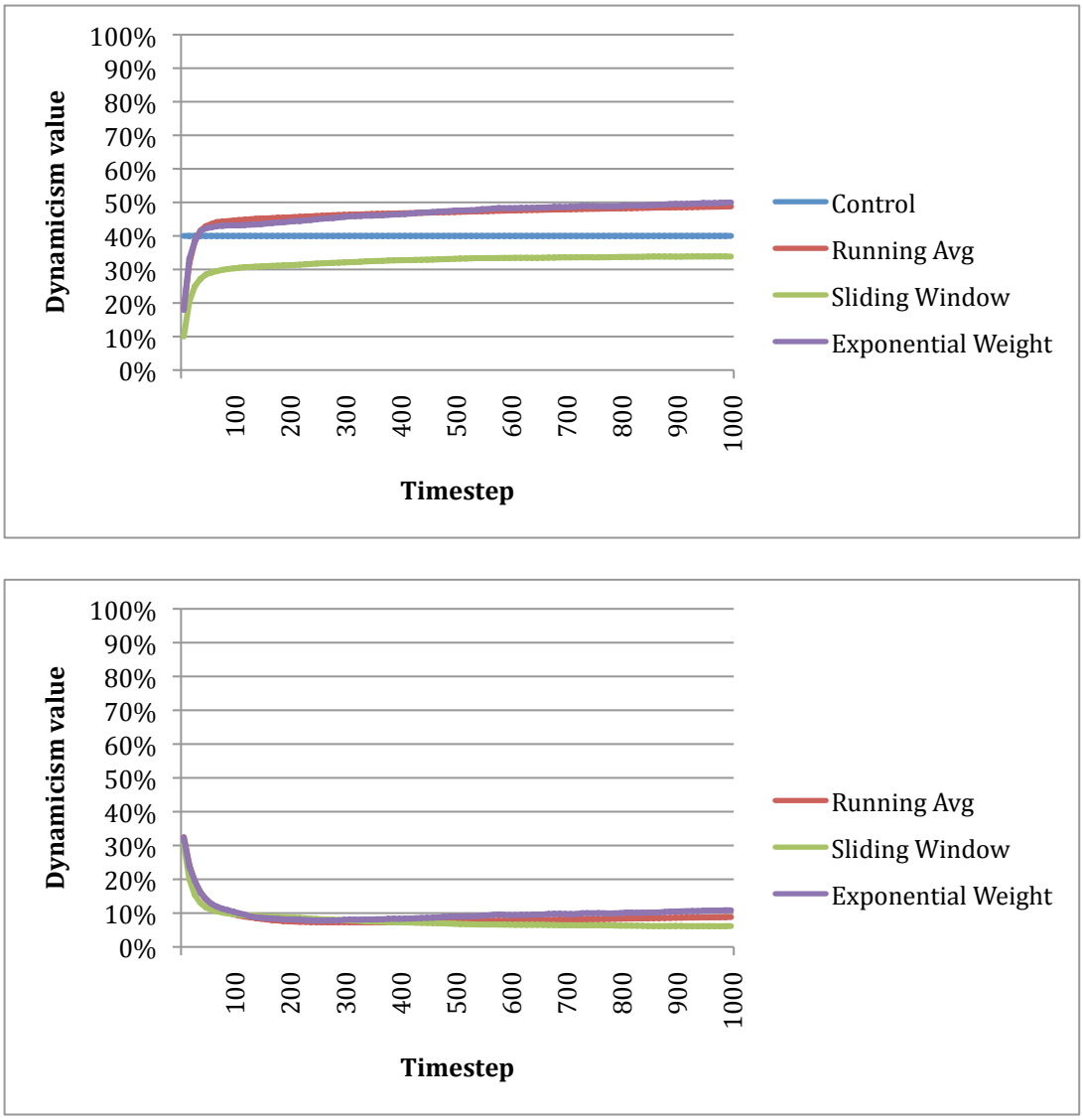
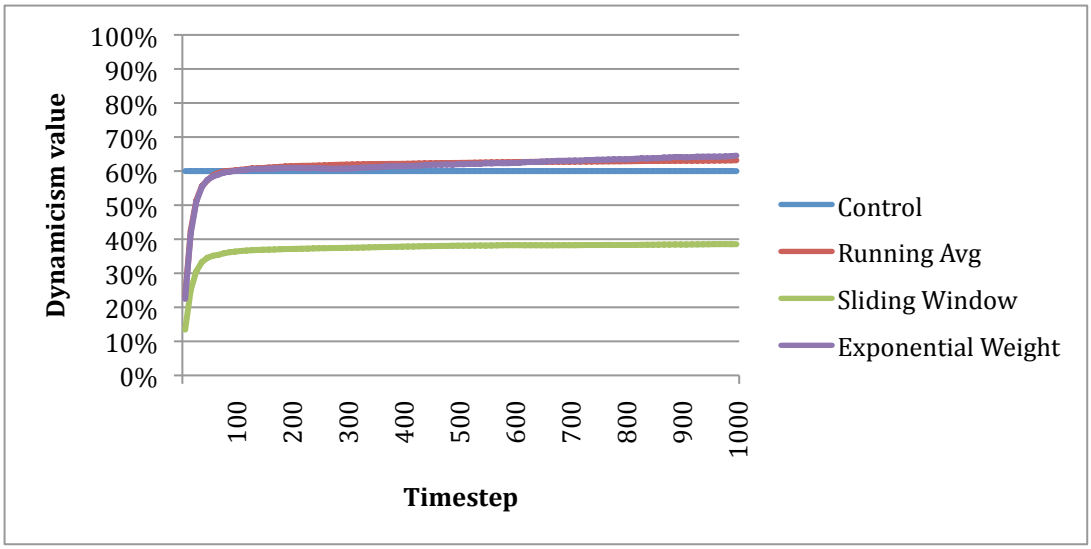


Figure 29a-b: EU learning strategies average estimation of 40% control dynamicism level and average estimation error



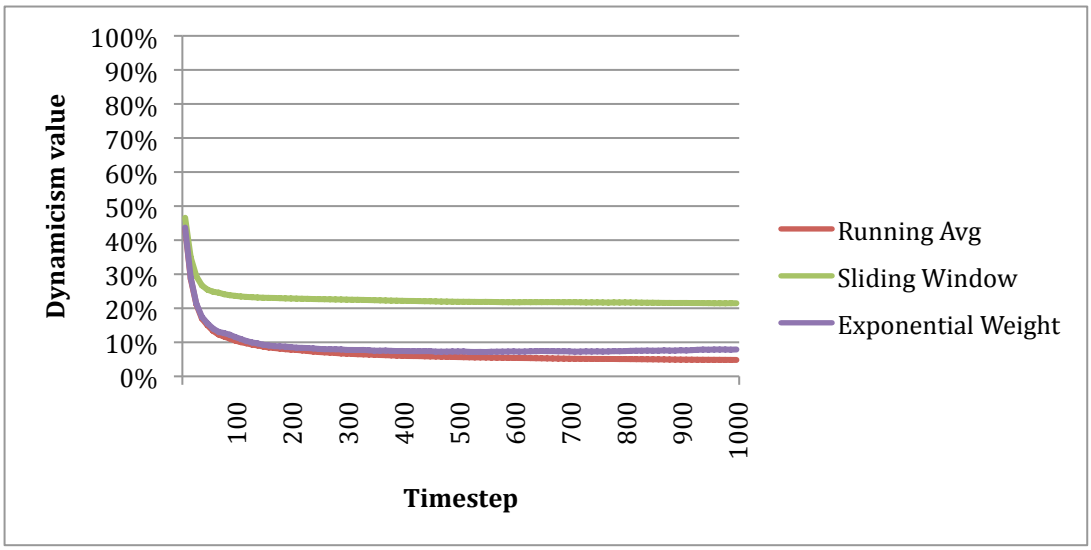
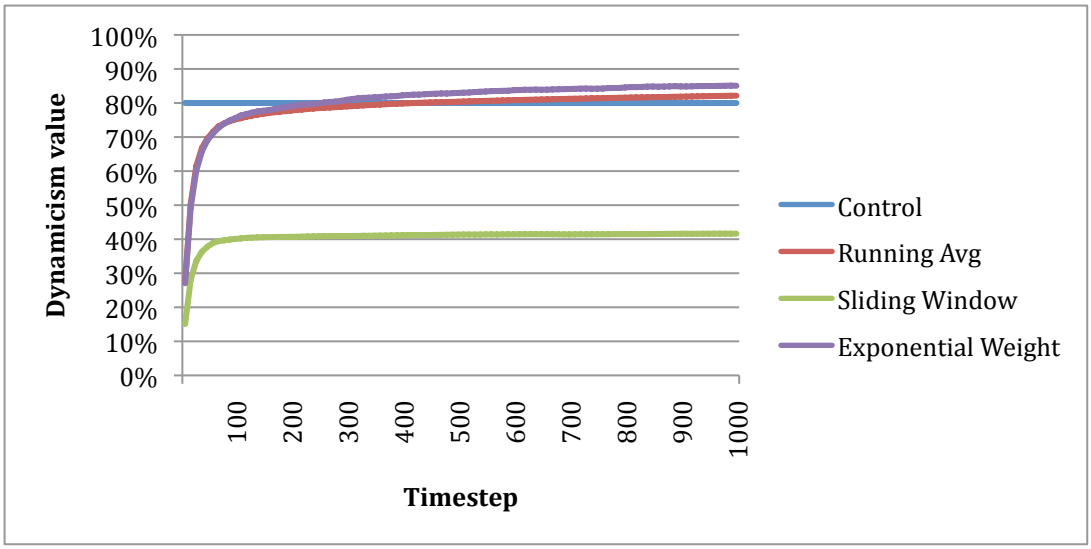


Figure 30a-b: EU learning strategies average estimation of 60% control dynamicism level and average estimation error



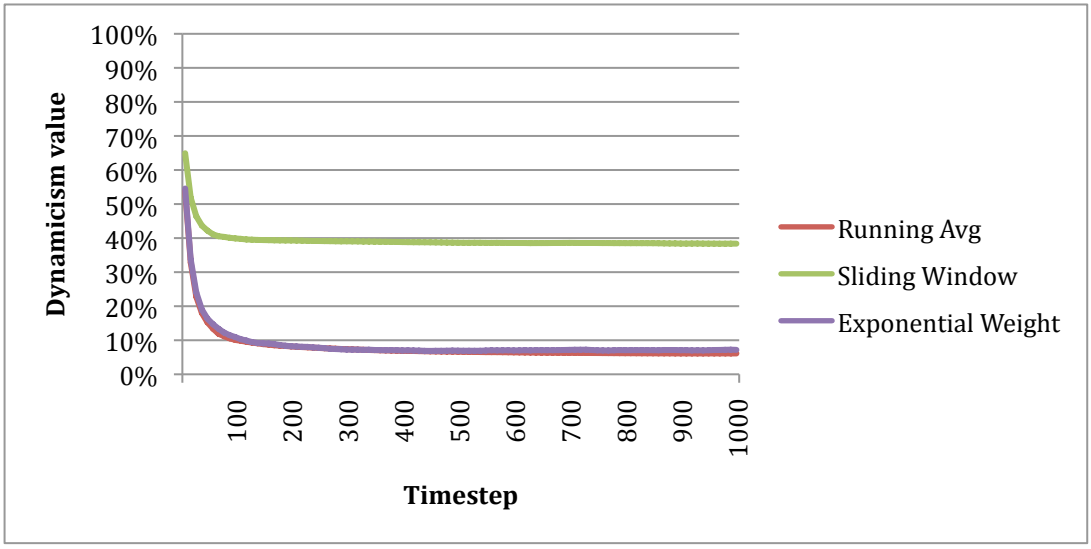
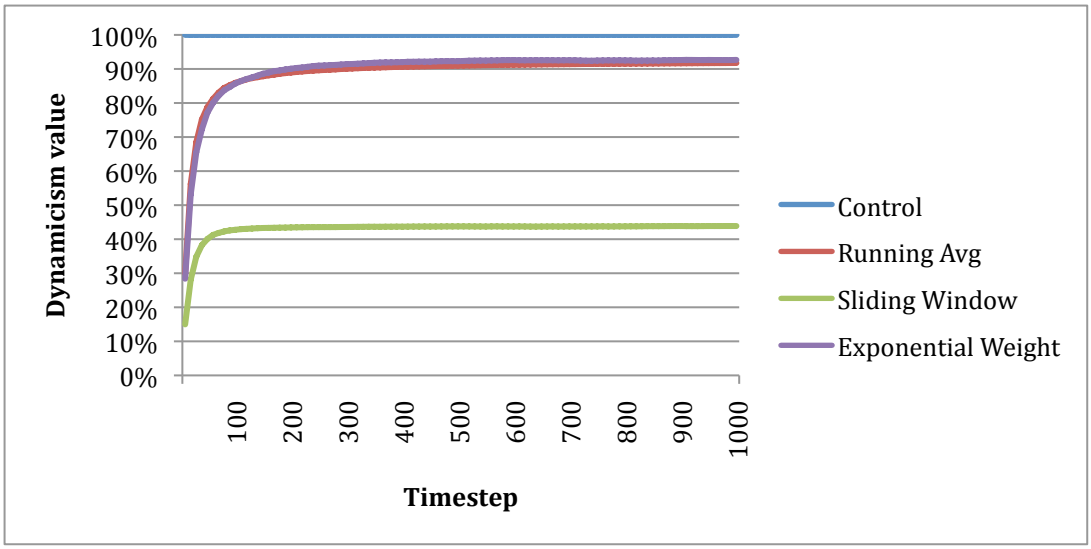


Figure 31a-b: EU learning strategies average estimation of 80% control dynamicism level and average estimation error



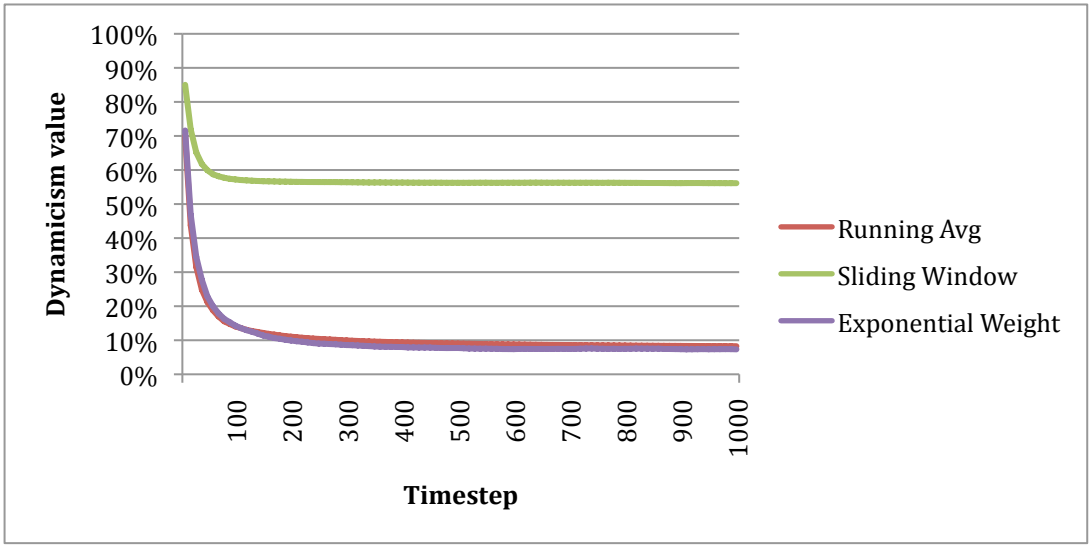


Figure 32a-b: EU learning strategies average estimation of 100% control dynamicism level and average estimation error

As can be seen from Figures 28 through 32 above, the Running Average and Exponential Weighting mechanisms performed well, usually converging to within plus or minus ten percent of the actual value of D within roughly 100 timesteps. Interestingly, both strategies tended to overestimate the value of D when D was at 20% and 40% of tasks changed during the lifetime of the job, as shown in Figures 28 and 29, and to

underestimate D when D was at 100%, as seen in Figure 32. However, as shown in Figures 26 and 27, these discrepancies did not have a highly substantial effect on the earnings of each strategy, and could probably be further addressed through further refinement of Equations 25 through 28 in Chapter 4.

In contrast, the Sliding Window strategy performed poorly, almost always converging to an estimated value of D of roughly 40% tasks changed during the lifetime of the job, regardless of the actual value of D. This is likely because the value in the figures above is the average value of hundreds of Sliding Window agents over ten different trials, whereas each individual Sliding Window agent at a particular instant in time would likely have had a wildly different idea of what D was, ranging all over the spectrum, as shown by the relatively high error estimates in the "b" figures. (Note that, as can be seen from the Running Average and Exponential Weighting results, even with an actual value of D at 100% in Figure 32, the agents do not generally make estimates of more than 90%. Thus, an average estimate error of around 40% implies that many agents could have seen, for example, an 80% actual value of D as being anything from 90% to 0% - this would still have resulted in an average estimation of roughly 40%, and an average error of roughly 40% from the true value of D.)

One possible explanation for this lack of performance is that the window size of the Sliding Window algorithm is too small – that it leaves the strategy with too few samples to make an accurate estimate. However, given that this window already represents 10% of the experiment's lifetime, and given that both the Running Average and Exponential Weighting strategies converge on the value of D within 100 timesteps, it seems more reasonable to conclude that, in this domain, completely abandoning all prior observations of D taken before an arbitrary point is a poor strategy, and that these observations should be, at most, slightly discounted (as by the Exponential Weighting strategy) rather than completely abandoned.

Figure 33 shows another comparison of an EU strategy with Known Dynamicism to the three EU-based learning strategies, in this case with each job in the experiment set to a different value of D randomly assigned from a level distribution between 0% and 100%. However, as can be seen from Figure 33, all strategies performed roughly the

same, as confirmed by ANOVA analysis, which finds that there is no statistically significant difference in the earnings.

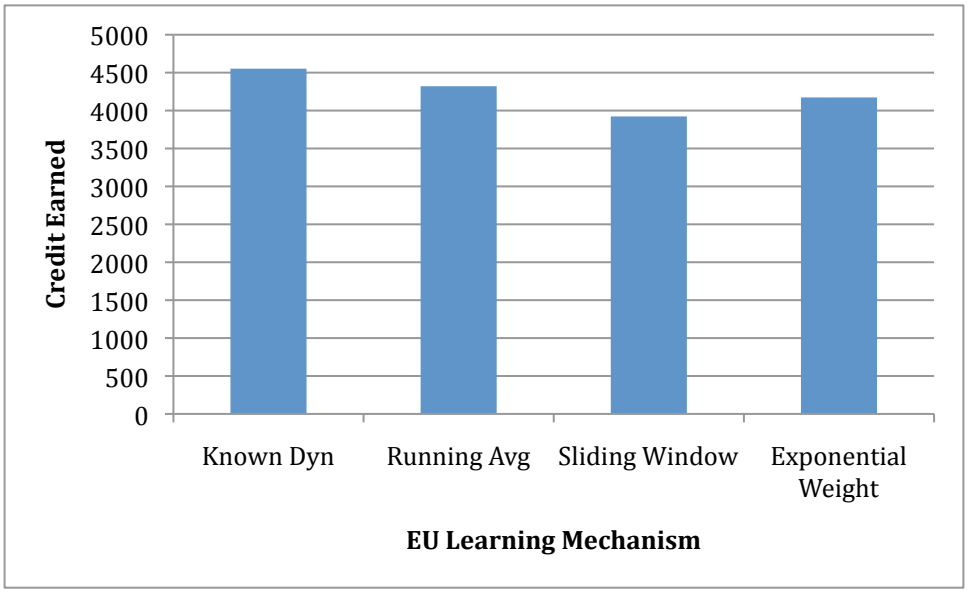


Figure 33: Average total earnings for EU learning strategies and EU with known dynamicism under random conditions

Lastly, Figure 34 shows a final comparison of an EU strategy with Known Dynamicism to the three EU-based learning strategies. Here, the control value of D was abruptly stepped twice during the experimental trials, starting at 25%, jumping to roughly 65% one-third of the way through the trials, then jumping back to roughly 35% two-thirds of the way through the trials. Figure 35 shows the evolution of D throughout the experiment, as well as the estimate of D for all three learning strategies.

As can be seen from the figures below, the earnings of the Known Dynamicism and Exponential Weighting strategies are essentially the same, while the earnings of the other two strategies are lower. In fact, ANOVA analysis indicates that there is a significant difference between the four strategies, but a t-test of the Known Dynamicism and Exponential Weighting strategies finds no significant difference between the two.

This observation is confirmed by the way that the Exponential Weighting strategy closely tracks the control value of D in Figure 35, immediately jumping when D changes

and slowly moving to a close approximation of the control. In contrast, although the Running Average estimation does change the first time D changes, by the second time there are too many obsolete historical observations to be overcome, and the estimation of D remains constant. Likewise, the Sliding Window strategy shows some small movements as D changes, but generally hones in on the roughly 40% estimation level seen in previous experiments.

Therefore, based on these results, it is apparent that the Exponential Weighting strategy performs best of the explored alternatives, to a point where it is statistically identical to an outright knowledge of the value of D in many instances.

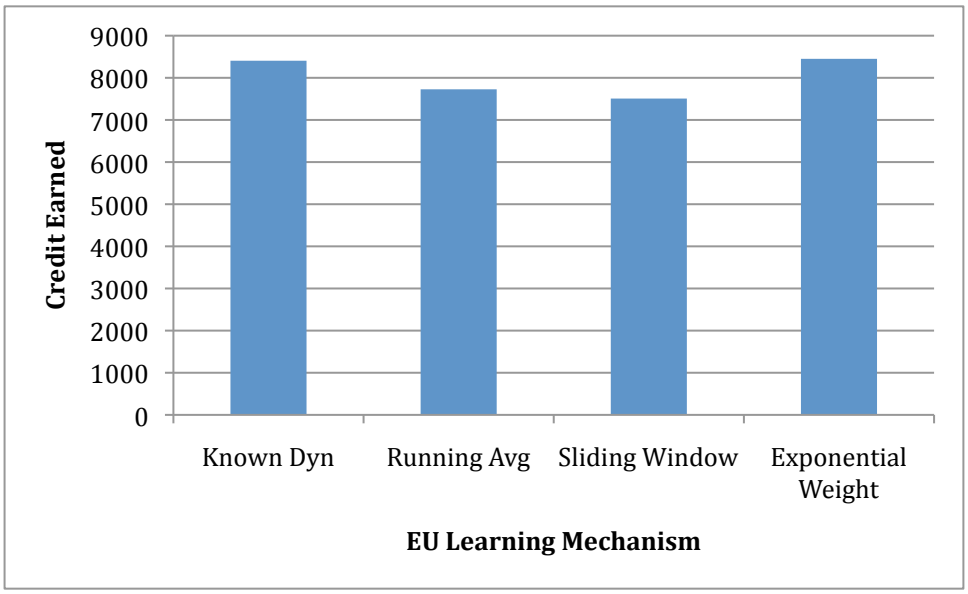


Figure 34: Average total earnings for EU learning strategies and EU with known dynamicism under step conditions

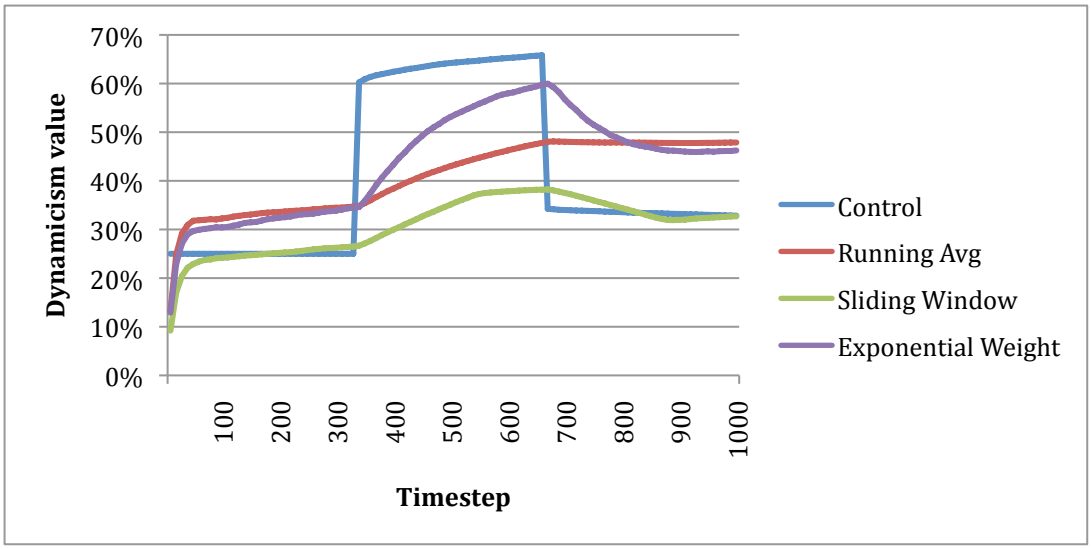


Figure 35: EU learning strategies average estimation of control dynamicism

Chapter 6: Robust Team Formation in the Consulting Domain

The experimental results laid out above in Chapter 5 have shown that an expected utility-based strategy can successfully form robust teams in a generic dynamic and contractless environment. However, as Research Question 5 states, "Can agents acting in accordance with the utility theory found in RQ2 be usefully deployed in a real world-based consulting domain?" More specifically, how does the EU strategy developed in this dissertation compare to existing team- and job-selection strategies prevalent in the real world consulting domain?

This chapter compares the performance of the expected utility-based strategy examined above to a "Manager" strategy embedded in a simulated consulting domain. More particularly, we wish to compare the performance of agents executing the expected utility-based strategy, either as a homogenous group or as part of a heterogeneous mix of agent strategies, against an agent strategy that enjoys many of the advantages that a manager at a large consulting company might enjoy – i.e. the ability to ensure that agents do not defect from an assigned task, and the ability to select agents that are currently idle, rather than having to recruit agents who may be otherwise engaged. In comparison, an agent executing an expected utility-based strategy is expected to be better able to predict and handle task instance changes and team member defections, and to have lower overhead compared to a centralized managerial organization.

Section 6.1 of this chapter provides an overview of the consulting domain, and compares and contrasts various aspects of the consulting domain with existing features of the simulation domain previously described in Chapter 4. Section 6.2 describes the Manager strategy and other aspects of the consulting domain simulation experiments, while section 6.3 provides the results of those experiments, as well as analysis of the relative performance of the expected utility-based strategy and the Manager strategy.

6.1 Consulting Domain Overview

Consultants are a broad class of domain experts and/or service providers who assist businesses and organizations with queries and tasks that are outside of their core capabilities. Consulting firms such as Accenture or IBM Global Technology Services are capable of assembling teams of experts in various fields to assist with various challenges a business might be facing, such as refining or reorganizing business processes, or implementing technological solutions to improve business efficiency (Hamm 2008; Accenture 2009). In addition, individual experts and service providers may act as freelance consultants, either as their primary career or as a supplement to their primary job (e.g. a university professor acting as a consultant in areas related to their primary area of research.) Consulting is a large and very lucrative industry; Kennedy Information Inc. estimated global consulting industry revenues were estimated to be 310 billion U.S. dollars for the year 2007 (Plunkett 2008).

Such freelance consultants essentially find work on an ad hoc basis, scheduling jobs as opportunity arises via informal communication networks. In contrast, the corporate consulting world has recently begun to make great advances in modeling worker skills, characteristics, and availability to allow for more efficient assembly of consultant teams to tackle different problems. For example, IBM has embarked on a project to model the expertise of its technical consultants to determine how best to deploy them on disparate projects (Baker 2008). Via internal software called IBM Professional Marketplace, project managers can find candidates who meet the required qualifications for a job, including characteristics such as technical or business skills, work visa status, availability, worker cost, organizational alignment, or job role. In IBM's case, this information is input into the system by integration of various databases, including human resources information, management chain information, and resume data (Heise 2009).

In addition, other companies are capable of mining similar information from other sources. For example, Microsoft's SharePoint Server is capable of mining email distribution lists and other social networking features to determine organizational attributes, while the Semantic Web is a broadly applicable metadata technology capable

of identifying vast amounts of information available via both internal networks and the World Wide Web (Berners-Lee, Hendler et al. 2001; DeBruyne 2009). Furthermore, these expertise identification technologies can, in some cases, be linked with sophisticated project management software such as Microsoft Project Server (Microsoft 2009), or can be used in conjunction with software development and modeling software such as IBM's Rational suite (IBM 2009).

Although these technologies are currently primarily employed by large, centralized consulting firms, it should be possible to duplicate many of the advantages these technologies bring – expertise identification on a global scale, professional-grade project modeling capabilities – to the wider world via open source technologies, much like open source projects such as Open Office have largely duplicated software packages such as Microsoft Office.

With such tools, freelance consultants would be able to identify and schedule work on jobs and with partners in the same way that centralized consulting firms do. The primary difficulties would be in getting agents to work together with potentially untrustworthy agents in an open environment such as the Internet, and building teams capable of adapting to the uncertainty and dynamicism inherent in a complex project. Dealing with these issues is, of course, the primary aim of this dissertation.

However, even given the importance of the consulting domain, the question still remains whether the consulting domain is a good match for the robust team formation strategies presented herein. To answer this question, we compare the two domains from two different perspectives below. First, we begin by listing characteristics of team formation between selfish agents operating in a dynamic, contractless environment, and ask whether these characteristics are commonly found in the consulting domain. Next, we list various characteristics of the consulting domain, and determine whether these characteristics can be integrated into the team formation simulation environment described above in chapter 3.

6.1.1 Characteristics of robust team formation environment

The first, most obvious aspects of the robust team formation environment described above in Chapter 3 are the **multiple selfish agents** attempting to complete **multiple jobs**. It is immediately apparent that the consulting domain possesses both of these characteristics – there are clearly numerous freelance consultants with many different areas of expertise in the world, and many consulting groups, both large and small, with numerous individual consultants at both the associate and partner levels. These consultants work on a wide variety of projects, either with the goal of maximizing profit for themselves or improving their position within a firm. Accordingly, it is reasonable to argue that the consulting domain features selfish agents working on different jobs for profit.

In addition, consultants collaborating via the Internet must frequently deal with the **contractless environments** found in the simulation environment, for multiple reasons. First, any collaboration via the Internet faces the threat of anonymity or fake identities – it is nearly impossible to enforce a contract when the identity of one or multiple parties is unknown or unreliable (Yokoo, Conitzer et al. 2005). Second, even when the identity of both parties is known, questions of jurisdiction arise. It is possible that the collaborating parties are in entirely different countries, in which case there may be no controlling legal authority capable of enforcing any contractual agreement, especially when one of the parties purposefully tries to break the contract. Finally, even when the identity of both parties is known, and both are in a common legal jurisdiction, it is arguable that the expense and effort involved with drafting and enforcing contracts largely negates many advantages – such as speed, ease, and cost – associated with collaborating via the Internet.

It is also worth noting that even consultants working together in a firm or corporation, where verified identities and legal jurisdiction are clearly not issues, may face some of the issues involved with failure of consultants to live up to stated obligation. More specifically, even employees at a firm may suddenly quit, fall ill, or otherwise be incapable of providing an expected skill or service. Such difficulties are clearly less than

those faced by online collaborators, but they do suggest that the research discussed in this dissertation may have some degree of applicability even beyond online collaborators.

Almost by definition, any job that requires a team of workers to complete it is **composed of multiple tasks**, frequently because different **individuals with distinct and specialized skills** are required to complete different aspects of the job. Certainly the consulting domain fits these criteria, since consultants are, again almost by definition, providers of specialized skills and expertise hired for the specific purpose of solving an uncommon problem that a business or other organization is not suited to handle. The simulation environment described in this dissertation clearly also captures this aspect of the consulting domain, not only with respect to technical or professional skills such as coding or business analysis, but with respect to secondary skills and characteristics as well, such as the ability to speak a specific language, geographic location and legal right to work in a specific country.

Our simulation environment and expected utility-based strategy also model **dynamic environments**, environments where the requirements to complete a job change in an unpredictable manner over time. The consultancy domain might face such discrepancies between the initial requirements for a job and the final requirements for a job because of several reasons. First, jobs can be inherently dynamic, with external circumstances forcing requirements changes at any time in the project development (Jalote 2001). Alternatively, jobs requirements may not change over time so much as be initially misunderstood, either due to incomplete information, bounded rationality or inexperience on the part of the requirements modelers. These changes to requirements are arguably not so much aspects of real environmental dynamicism so much as inaccurate perceptions of job requirements, but regardless, the strategies described above for providing adaptable teams that can handle unexpected job requirements are workable solutions.

Finally, the paramount decisions in the simulation environment described above are **what jobs should an agent work on**, and **what agents should be partnered with**? These considerations are particularly important for freelance consultants who face an open marketplace full of opportunities to pursue, and who must judiciously determine which opportunities are likely to be most profitable, and what teams are most likely to be

able to complete a job. Again, the work presented here is ideally suited for allowing these consultants to determine what jobs to pursue and what partners to work with in order to maximize profit.

6.1.2 Characteristics of consulting domain

Once we have determined that various aspects of the simulation environment described in Chapter 3 are relevant to the consulting domain, we must ask whether various aspects of the consulting domain can be modeled by the simulation environment. Section 6.1.1 has already addressed some characteristics of the consulting domain, such as agents with different skills of various kinds – technical, professional, interpersonal – and also different qualifications and attributes, such as geographic location, or permits or visas to work in different countries. All of these characteristics can easily be modeled by the simulation environment by making these skills and characteristics individual tasks (e.g. a Java coder is a type of task, and the ability to speak Spanish is a different type of task, both of which are initially active) or by making combinations of these skills and characteristics an individual task (e.g. a Spanish-speaking Java coder is a single task, and is distinct from a French-speaking Java coder, or a Spanish-speaking C coder).

In addition to having consultants with different skills, the real-world consulting domain features **different competencies of the same type of skill**. For example, while there are many C++ coders in the world, some are beginners, some have a moderate level of expertise, and some are true experts. It is also the case that **jobs and tasks require different skill levels** – the same basic task of writing Java code, for example, can be relatively simple (a small utility application), somewhat difficult (writing application code as part of a team) or very difficult (architecting a distributed application). These difficulties can, in general, be matched to different skill levels, so that experts do difficult tasks and beginners do simple tasks. Furthermore, while someone of a higher skill level is generally able to accomplish the same work as someone of a lower skill level, the reverse is not necessarily true.

In the simulation environment, this is mirrored by the concept of quality-of-service levels in tasks and agent skill levels, where different agents have the ability to solve some

tasks, but not others, of the same type. Furthermore, just as expert consultants can frequently command a much higher fee than junior consultants, there is a substantial difference in the earnings that an agent completing a task with a high quality-of-service demand can expect, relative to an agent completing a low quality-of-service demand.

In addition to quality-of-service differences in task rewards, **different types of tasks are inherently more profitable than others**. In the software business domain, for example, it is the case that administrative assistants, accountants, and coders are all needed to make a working software company, but because of various factors, working on coding tasks is inherently more profitable than working on accounting or administrative tasks. Likewise, our simulation has a value multiplier associated with each type of task, in addition to making longer tasks (e.g. tasks requiring more effort) worth more than shorter tasks. The simulation environment is therefore able to capture a great deal of variability in the expected payoff for tasks, from the expensive (e.g. a long tasks with a high task value multiplier and a high degree of difficulty) to the inexpensive (e.g. a short task with a low value multiplier and a low degree of difficulty). Accordingly, although the simulation does not directly model the explicit fees that a consultant can demand (e.g. an experienced lawyer might ask for several hundred dollars an hour to work as a consultant), more skilled agents can effectively demand a higher reward from the market by electing to work on jobs where they are assigned more profitable tasks.

An additional characteristic of the consulting domain is that **different consultants are available at different times** – when a consultant is working on a given project, his or her ability to take on new work will be at the very least, diminished, if not outright eliminated. For the purposes of experimentation, we assume that agents can only work on one project at a time, although it would certainly be possible to modify the simulation to behave otherwise. In addition, our previous experiments have made the assumption that agents must communicate via message to determine availability and interest between foremen and worker agents – this is arguably a worthwhile assumption when communicating between agents on the Internet, who may be able to communicate instantaneously but may also take additional time to consider one or more offers to join a team before replying. In contrast, consultants who work for a large organization may

have their availability made far more explicit to other partners via a centralized scheduling system (Microsoft 2006). Accordingly, as will be described in further detail in section 6.2, we have adapted our experiments in part to remove the blind guessing about agent availability seen in previous experiments.

Lastly, it is the case that **complex, multi-task jobs are often characterized by uncertainty as to the duration of both the project as a whole, and of individual subtasks** (Shtub, Bard et al. 2005). However, our simulation work, while certainly allowing for uncertainty in the total duration of entire jobs due to changing task requirements and changing team membership, does not allow for tasks to be of variable duration. This is largely because other work has already explored how to deal with uncertainty in task duration (Ahn, DeAngelis et al. 2007; Stein, Payne et al. 2007), and determining how to integrate the robust team formation strategies described here with their work is probably best left for the future.

6.2 Consulting Domain Experimental Setup

To better test the expected utility-based strategy in real world conditions, we compare the previously described EU strategies to a new **Manager** strategy, which is designed to more closely approximate the arrangement of real world consulting firms. More specifically, the Manager strategy acts the same way a project manager in a centralized consulting firm does, in that it is responsible for selecting a job and assembling a team of agents to work on the job, but does not actually work on any of the component subtasks of the job itself.

The Manager strategy in these experiments also has many of the advantages that a manager in a centralized firm does, in that it not only commands the loyalty of agents (i.e. agents cannot quit working for it until they have finished their assigned tasks), but that it knows the availability of potential partners, and will not waste time trying to recruit potential partners who are otherwise engaged. Additionally, the Manager strategy will never take the initiative to look for a more profitable job while it is already working

on a job, but will always look for a new job to work on when it is idle, rather than waiting for an offer from another agent.

(Note that, in comparison, all other strategies, including the MasterOracle strategy, do not know if the offers it makes to potential partners are being directed towards agents who are already working on a job, and will, at random times, look for other jobs to work on as a foreman, even if it is already occupied.)

The Manager strategy is experimentally compared to the EU strategy in two different contexts. In the first, the EU strategy is part of a mixed group of agents, with all other agents following the heuristic strategies described above in Chapter 3. In the second context, all agents in the experiment follow the EU strategy. All experiments are carried out with the experimental parameters from Table 2.

6.3 Consulting Domain Experimental Results

Figure 35 shows the average per-agent earnings for all agents in the Manager experimental trials, as compared to the average total earnings for agents following the EU strategy in the two alternate contexts. Figure 36 provides an alternate view of the same data by showing average EU earnings as a percentage of the average earnings for each agent operating under the Manager strategy.

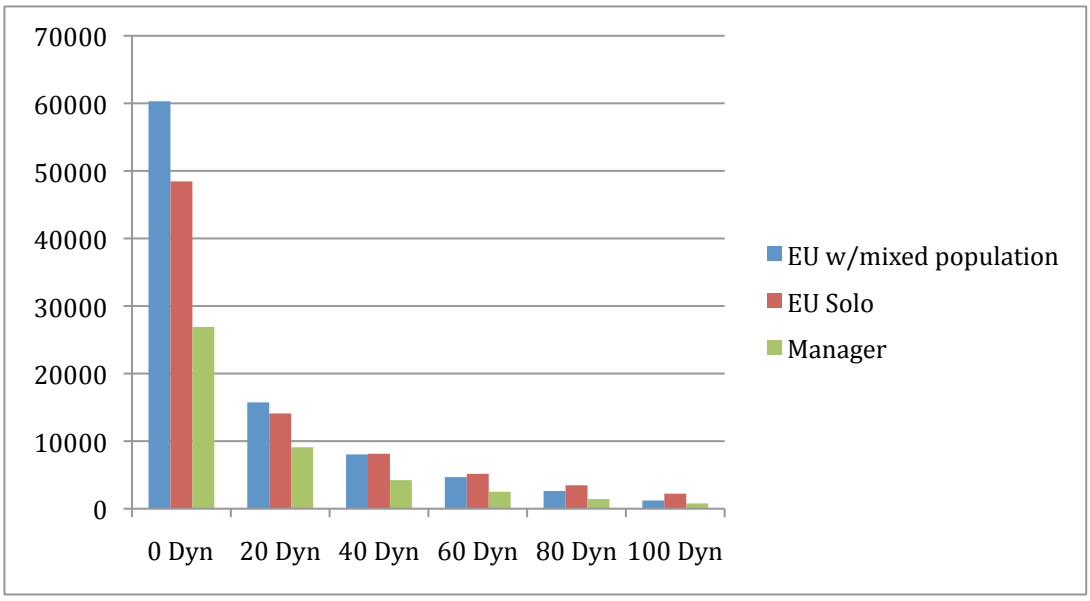


Figure 36: Average per-agent earnings for all agents under Manager strategy vs. average EU earnings

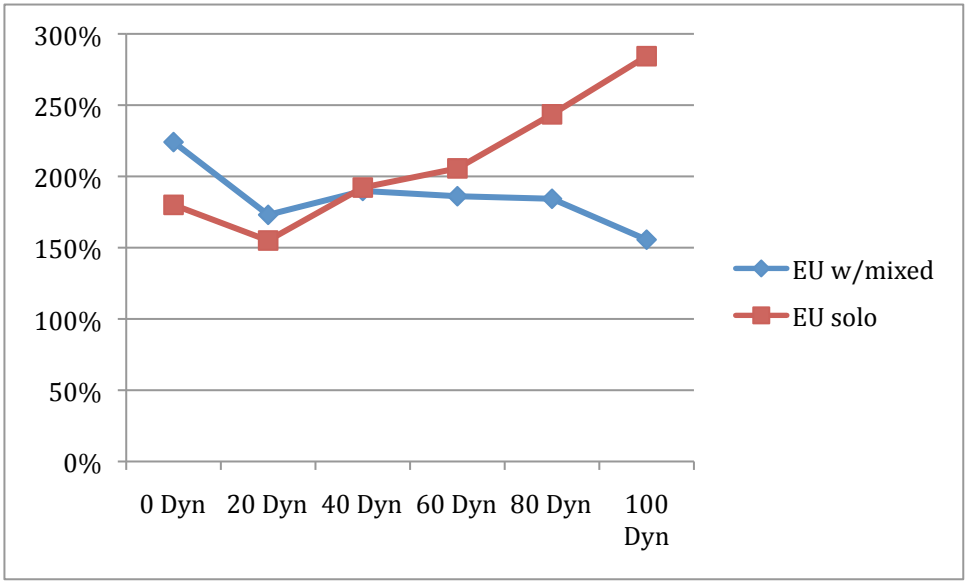


Figure 37: Average EU earnings as a percentage of average per-agent earnings under Manager strategy

As can be seen from the graph, both of the EU strategies significantly outperform the Manager strategy, as confirmed by t-test comparisons of all strategies at all dynamicism

levels. This result strongly suggests that workers can earn more profit when they are allowed to pursue now opportunities as they arise, rather than being locked into a particular job until finished. The EU strategy which deals with a heterogeneous mix of unreliable partners still performs better than the Manager strategy, even in purely static environments, suggesting that the EU strategy's ability to handle changing job requirements is not the primary advantage it has over the Manager strategy.

Instead it seems likely that the EU strategy has an advantage in that each agent pushes for the most profitable job *for itself*, whereas the manager agent is only interested in maximizing total revenues and there is no mechanism for an agent to find its maximally profitable job. Rather, the Manager assigns agents more or less randomly to any task they can handle. This also explains the even higher profit in environments where all agents are using the EU strategy, since every agent seeks out (or takes new offers to work on) new jobs that better suit its abilities, rather than sticking with an assigned task.

Figure 38 through 40 offer supporting evidence for this argument. Figure 38 shows the total jobs completed by all agents in each experimental setting, whereas Figure 39 shows the total profit earned by all agents combined in each setting (unlike Figure 36, which shows the average profit per agent). Figure 40 shows the average worker profits per class of agent working under the Manager strategy, and is accompanied by a reprint of Figure 12b, which shows the same data for agents working with the Expected Utility strategy.

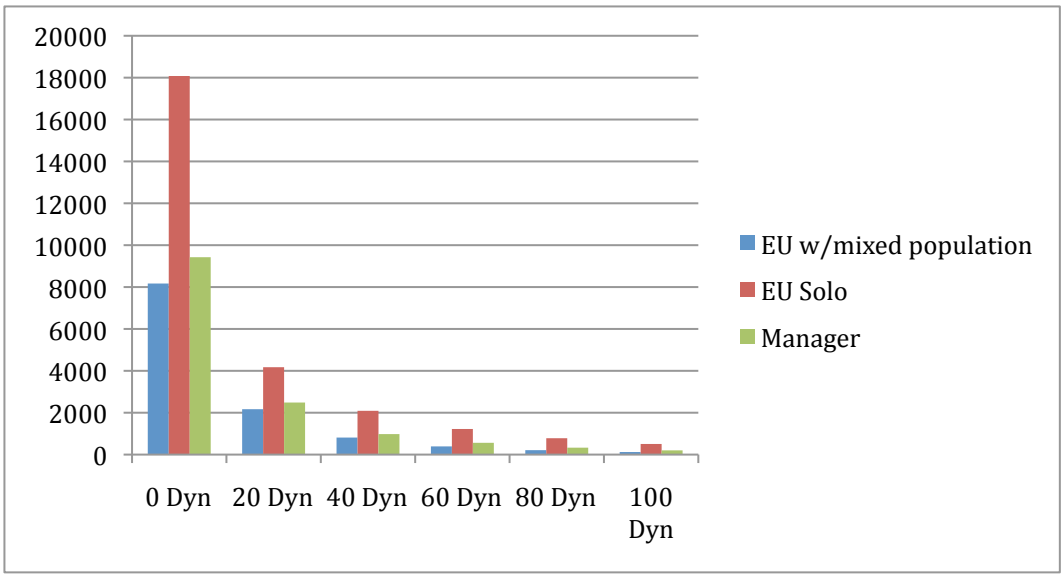


Figure 38: Total jobs completed for Manager strategy vs. EU strategies

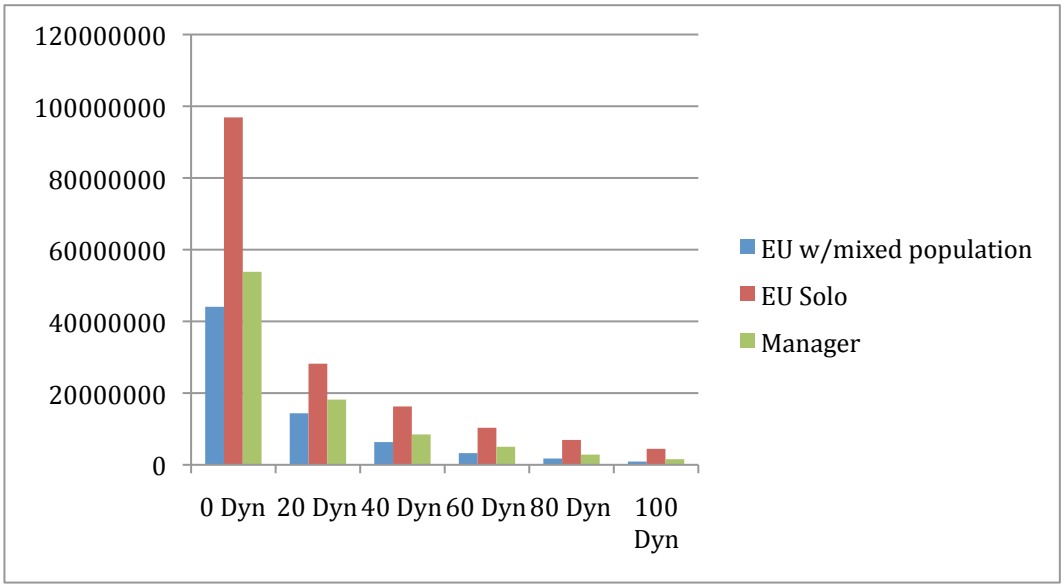


Figure 39: Total earnings for all agents for Manager strategy vs EU strategies

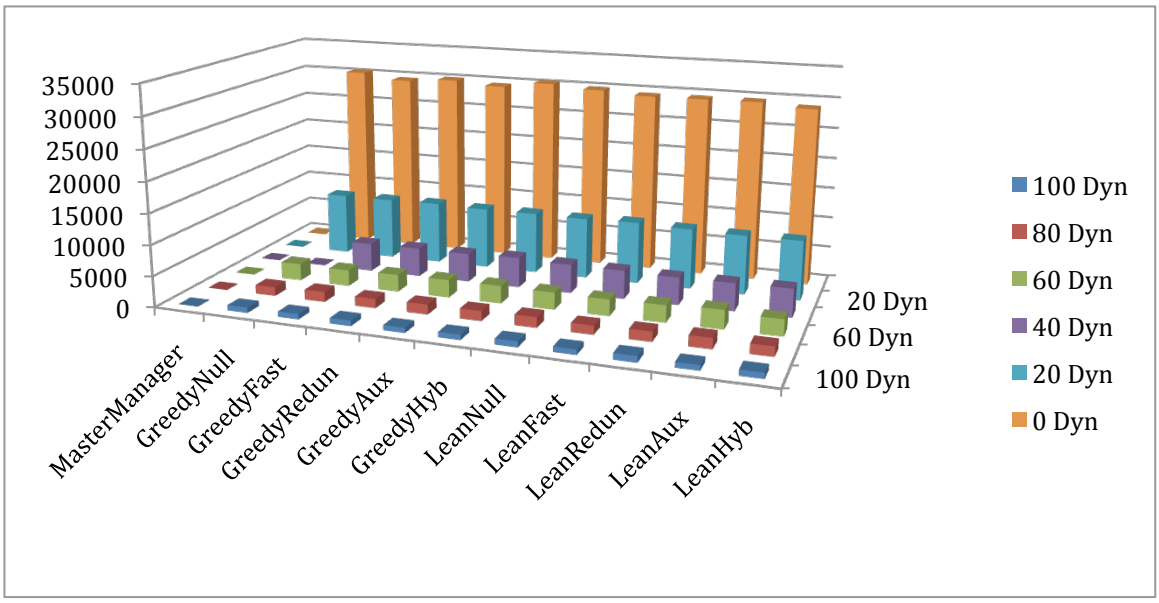


Figure 40: Worker earnings for agents under Manager strategy

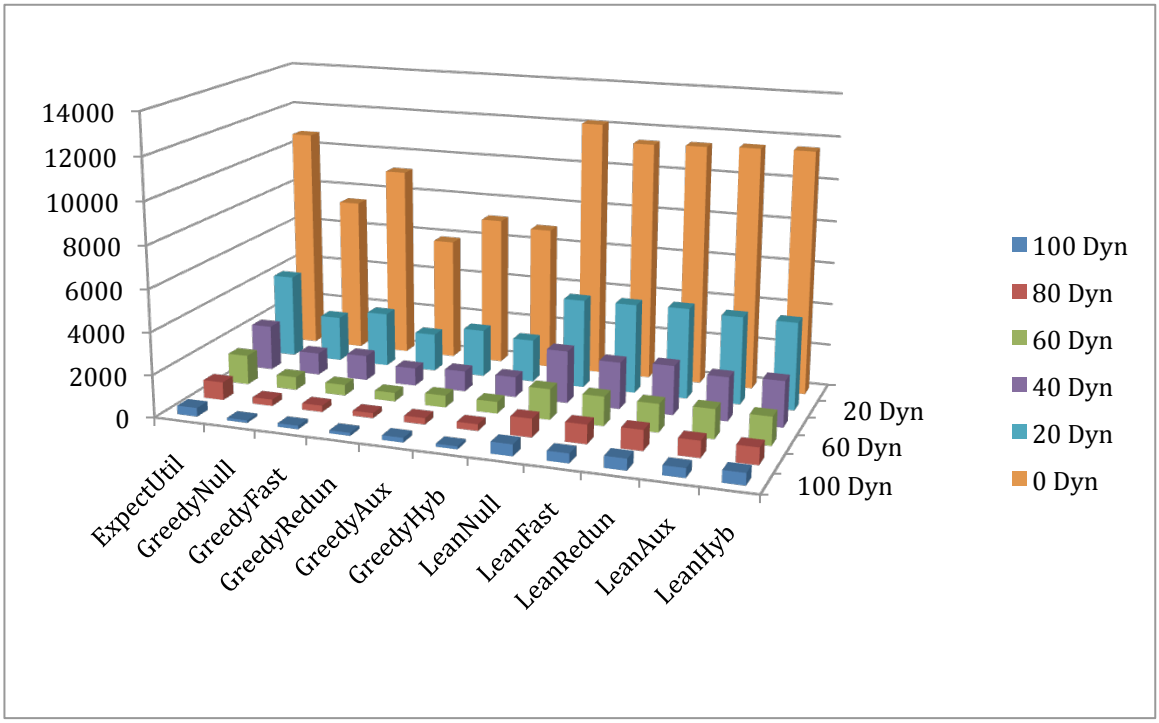


Figure 12b: Worker earnings for agents under Manager strategy

Specifically, Figures 38 and 39 are noteworthy because they show that the number of jobs completed and global profit earned is actually higher for the Manager strategy than for the EU strategy with mixed partners. (Note that these graphs are roughly identical, and total earnings generated for all agents is directly correlated with the total number of jobs completed by all agents.) EU agents earn far more than agents under the Manager strategy even though fewer jobs are completed; EU agents therefore have a higher profit per complete job. This is further proven by Figures 40 and 12b – note that individual worker profit is far higher and better distributed under the Manager strategy, whereas under the EU strategy, EU foremen choose jobs that most benefit them as individuals, rather than the group as a whole.

Figures 38 and 39 also give some indication as to why the Manager model is so widespread in real world domains. Note that, from a broad standpoint, the Manager model is actually a more productive model than an EU strategy operating with other mixed strategies. This is important for two reasons: first, because the Manager model actually does a better job at maximizing the productivity of an entire group, and second because the Manager model is actually better at completing more jobs, even if it is not as good as the EU agent at determining the profitability of each job.

This dissertation has largely ignored the beneficial effects of repeated interactions between satisfied buyers and sellers, and it is reasonable to argue that in an online, largely anonymous environment populated by increasingly transient virtual organizations, customer loyalty will be less important, or outright impossible to maintain. However, in the traditional business settings where relationships and repeat business are very important, it certainly has been important to keep satisfied clients, which explains the Manager strategy's traditional popularity.

Note also that experimental trials where all agents use the EU strategy were far more profitable and successful than other strategy pairings. However, this kind of setup would have been virtually impossible to set up in the past, when individual workers could not be reasonably expected to do their jobs *and* the in-depth calculations required to execute the EU strategy for job and team selection. Even today, when it is possible to imagine large

groups of freelance consultants each choosing jobs and partners based on the recommendation of a personal scheduling agent running the EU strategy, it is possible that there may be as yet unidentified incentives that would push individual freelancers to defect from a universally-followed EU strategy. It is therefore unclear whether a universal EU strategy is feasible from a game theoretic standpoint, and more research into this area is likely warranted.

Regardless, however, it remains the case that the EU strategy has proved more profitable than the traditional manager strategy. Note also that, although the manager strategy completes more jobs than the EU strategy in environments with a mix of agent strategies, the manager strategy does so only because it can command other agents to be completely reliable. In the open environments such as the Internet, where agents interact anonymously with no controlling authority available, the Manager strategy could not operate, and the worth of the EU strategy becomes even more apparent.

Chapter 7: Summary and Conclusions

This dissertation focused on robust coalition formation between selfish agents in a dynamic environment where contracts are unenforceable. Such environments are found in numerous domains, one in particular being freelance consultants collaborating over the Internet.

Previous agent research has covered different aspects of the problem of selfish agents operating in dynamic, contractless environments, but no research successfully addresses all of these aspects – selfish agents, dynamic and contractless environments - in combination. For example, although algorithms exist that are capable of efficiently partitioning selfish agents into teams assigned to given jobs in a static environment (Kraus, Shehory et al. 2003), such algorithms are inadequate to address the exponential number of potential job permutations in a dynamic environment. Likewise, although mechanisms exist for allowing teams to adapt to dynamic job requirements (Scerri, Farinelli et al. 2005), such teams are composed of selfless agents, and are not applicable to selfish agents which may not accept a loss of utility as tasks change. Finally, although contracting mechanisms exist that are capable of allowing selfish agents to deal with uncertain information or dynamic environments (Sandholm and Lesser 1996), such mechanisms are clearly not applicable to domains such as the Internet where contracts cannot be enforced. Therefore, because no research currently covers all aspects of coalition formation between selfish agents in a dynamic, contractless environment, a novel approach is required.

The main hypothesis of this dissertation is that, *in a contractless, dynamic environment, peer groups of autonomous agents can form high-utility teams by forming robust teams that balance skill set adaptability and redundancy with individual agent incentive*. Accordingly, this dissertation accordingly has three major goals: to develop a theoretical framework that describes how selfish agents should select jobs and partners in

a dynamic, contractless environment, to test that framework against existing heuristics in a simulated environment, and to create a learning agent capable of optimally adjusting its coalition formation strategy based environmental factors.

The dissertation successfully addressed each goal. First, an expected utility-based strategy was developed based on a function that allowed different job/team combinations to be compared for expected profit. This EU strategy was described in theoretical terms, and placed inside a simulated environment for comparison against both pre-existing heuristic-based strategies and the newly developed Master, Oracle, and MasterOracle "ideal" strategies. These ideal strategies do not operate under the same unpredictably dynamic and contractless constraints as the heuristic and EU strategies, and accordingly provide a theoretical baseline for maximum earnings. The EU strategy was found to perform significantly better than pre-existing heuristic-based strategies, and was roughly within one order of magnitude of the ideal strategies in terms of average agent earnings per strategy.

More specifically, agents following the Expected Utility strategy earned at least one fifth the credit of agents following the "Oracle" strategy, which allowed agents to see into the future, which strongly suggests that the EU strategy does a reasonably good job of estimating the true value of a job in a dynamic environment. The EU strategy also compared well to the "Master" strategy, which prohibited partner agents from defecting. In highly dynamic environments, the EU strategy even outperformed the Master strategy, which strongly suggest that the risk of building teams of defection-prone agent can be offset by creating flexible teams that can deal with both defections and changes in job requirements. The EU strategy is also found to work well under different environmental conditions where agent skills are plentiful, or jobs are scarce, but that none of the presented strategies, including the "MasterOracle" strategy performed well when the number of skills required to successfully complete a job was high compared to the number of agents available. Further testing also indicated that the EU strategy is able to accurately estimate the earnings from a given job and the probability of success for a given job, but that further refinements on the EU strategy's estimation of a job's length may be useful.

Learning mechanisms were also developed which allowed the EU strategy to detect the current dynamicism value of the environment, rather than being given the precise value in advance. A learning mechanism using exponential weighting of observations was found to be highly effective, and statistically indistinguishable from an EU agent operating with a precise, known value of dynamicism. The learning mechanism was also able to quickly track changes in the level of dynamicism found in the environment (i.e., how much of a job's requirements changed during the lifetime of a job) and to adjust agent strategy accordingly.

This research also considered the real world domain of the consulting industry analyzing the similarities and differences between the consulting domain and the existing simulation environment developed for prior work. Based on this analysis, changes were made to both the simulation environment and the theory underlying that environment to allow for new factors to be considered, such as task quality of service and precedence between tasks in a job. Furthermore, the EU strategy was tested against a "Manager" strategy designed to mimic the internal workings of a large, centralized consulting firm. The EU strategy was found to be more successful at earning profit than agents working under the Manager strategy, both when working with other, heterogeneous strategies and in a homogenous, EU-only group of agents.

These results strongly imply that the EU strategy would be highly valuable operating in real-world environments where job requirements can be expected to change significantly over the lifetime of a job, and where contracts between humans and/or agents cannot be enforced, such as freelance consulting on the Internet. Furthermore, by providing a strategy which is capable of dealing with selfish agents in dynamic, contractless environments, this dissertation fills a previously unexplored area of agent research, thereby extending the potential reach and value of all software agents.

7.1 Assumptions and Future Work

The EU strategy has been proven capable of forming high-utility teams in dynamic, contractless, environments, and of significantly outperforming existing heuristic-based strategies. The EU strategy is based on an environmental domain assumed to have certain properties:

- **Agents are not limited by knowledge about prospective jobs, partners, requirements and skills.** This is believed to be a feasible assumption, based on the current power and potential reach of modern search technology and the Semantic Web.
- **Agents earn profit directly proportional to the tasks they complete.** This is partially because negotiated profit sharing lies beyond the scope of this research, and partially because enforcing division of profit sharing implies enforcing profit sharing agreements, which runs counter to the basic idea of a contactless environment.
- **Agents not working on an assigned task are assumed to have defected.** This is a simplifying assumption based on the clear need for abandoned tasks to be quickly reassigned to another team member – note that determining whether a teammate who works only "part time" on a given task has actually quit, or is simply working on a different team is far more complex than simply noting that a teammate has not performed its required action. However, future work could address this issue by specifying that agents must spend an agreed-upon percentage of time working on a specific team, and altering the EU calculations accordingly.
- **There is a direct 1-to-1 mapping between skills and tasks.** This is again a simplifying assumption that makes it easier to determine what tasks are covered and what are not. However, future work could likewise address this assumption by defining a task as a complex demand for multiple agents to simultaneously use a given skill to complete the task, rather than simple 1-to-1

mapping of tasks and skills. This, in turn, would likely necessitate further changes to the core EU algorithm.

- **The supply of workers and skills is fixed per simulation.** Furthermore, because the price structure of agent rewards for completing certain types of tasks is fixed and randomly generated in a manner independent of the actual distribution of tasks and skills, this dissertation does not address any supply-and-demand type questions that might show how the supply of workers and skills might change over time due to price signals. This is likewise work that might be addressed in the future.

In addition, further modifications to the EU strategy itself are possible, both to handle new environmental features (e.g. more precise information on the likelihood that specific tasks will change, or that specific agents will defect) and to further refine existing features (e.g. improving the accuracy of estimating D). In addition this work could be integrated with general trust and reputation algorithms by allowing the EU strategy to act as an interim strategy during the "bootstrap" period of reputation learning. More precisely, the EU strategy could operate with starting estimate for general agent reliability during periods where existing trust and reputation algorithms are still determining the reliability of individual agents, and gradually be phased out or integrated with other trust and reputation strategies as reliability information becomes more certain.

Additional work could also be done on adapting the EU strategy from an "anytime" algorithm that simply selects increasingly superior job/team combinations when faced with a random stream of potential opportunities, to a strategy that actively shapes an active search for the optimal job/team parings in an environment. This would allow the EU strategy to better compete in environments where there are a high number of skills required for a job, compared to the number of agents available to partner with. Lastly, the game theoretic aspects of agents choosing an EU strategy vs. other strategies in a mixed competitive/cooperative environment could be addressed.

Bibliography

Abdallah, S. and V. Lesser (2004). Organization-based cooperative coalition formation: Proceedings of the Intelligent Agent Technology, IEEE/WIC/ACM International Conference162-168.

Accenture. (2009). "Global Management Consulting, Technology and Outsourcing Services from Accenture." Retrieved 5/3/2009, 2009, from http://www.accenture.com/home/default.htm.

Ahn, J., D. DeAngelis, et al. (2007). Attitude Driven Team Formation using Multi-Dimensional Trust, Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IEEE Computer Society Washington, DC, USA: 229-235.

Andersson, M. R. and T. W. Sandholm (2001). Leveled commitment contracts with myopic and strategic agents, Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, Elsevier. 25: 615-640.

Baker, S. (2008). The numerati, Houghton Mifflin Co. Boston, MA, USA.

Berners-Lee, T., J. Hendler, et al. (2001). The semantic Web. Scientific American 284: 28-37.

Chalkiadakis, G. and C. Boutilier (2004). Bayesian Reinforcement Learning for Coalition Formation under Uncertainty, Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, IEEE Computer Society Washington, DC, USA: 1090-1097.

Corkill, D. D. and V. R. Lesser (1983). The use of meta-level control for coordination in a distributed problem solving network: Proceedings of the Eighth international joint conference on Artificial intelligence, 748-756.

Davis, M. and M. Maschler (1963). THE KERNEL OF A COOPERATIVE GAME, DTIC Research Report AD0418434.

de Weerdt, M., Y. Zhang, et al. (2007). Distributed task allocation in social networks. AAMAS. Honolulu, Hawaii, USA, ACM.

DeBruyne, D. (2009). Phone Interview with Drew DeBruyne.

Dellarocas, C. and M. Klein (2000). An experimental evaluation of domain-independent fault handling services in open multi-agent systems: Proceedings of the Fourth International Conference on MultiAgent Systems 95-102.

Faratin, P. and M. Klein (2001). Automated Contract Negotiation and Execution as a System of Constraints, International Journal of Human-Computer Studies, Volume 67, Issue 1. MIT, Cambridge.

Fullam, K. (2007). Learning Trust Decision Strategies in Emerging Reputation Networks. Electrical and Computer Engineering. Austin, University of Texas. Ph. D.

Gaston, M., J. Simmons, et al. (2004). Adapting network structures for efficient team formation, Autonomous Agents and Multi-Agent Systems , Volume 15 Issue 1.

Gaston, M. E. (2005). Agent-organized networks for dynamic team formation, Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, ACM Press New York, NY, USA: 230-237.
Guessoum, Z., J. P. Briot, et al. (2005). Towards Fault-Tolerant Massively Multiagent Systems. LNCS, Volume 3446/2005, Springer.
Gujral, N., D. DeAngelis, et al. (2006). Modeling Multi-Dimensional Trust. AAMAS 2006 Workshop on Trust in Agent Societies
Hamm, S. (2008). "International Isn't Just IBM's First Name." BusinessWeek.
Hanna, H. and A. I. Mouaddib (2002). Task selection problem under uncertainty as decision-making, Proceedings of the first international joint conference on Autonomous agents and multiagent systems ACM Press New York, NY, USA: 1303-1308.
Heise, S. (2009). "Phone Interview with Steve Heise."
IBM. (2009). "IBM Rational Software." Retrieved 5/3/2009, 2009, from http://www-01.ibm.com/software/rational/.
Jalote, P. (2001). Software project management in practice, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.
Johansson, S. J. (2006). On using multi-agent systems in playing board games, Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems ACM Press New York, NY, USA: 569-576.
Jones, C. L. D. and K. Barber (2008). Combining Job and Team Selection Heuristics. 2008 AAMAS Workshop on Coordination, Organization, Institutions and Norms. Lisbon, Portugal, ACM.
Jones, C. L. D. and K. S. Barber (2007). Bottom-up Team Formation Strategies in a Dynamic Environment: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology 60-74.
Ketchpel, S. (1994). Forming coalitions in the face of uncertain rewards, Proceedings of the twelfth national conference on Artificial intelligence, American Association for Artificial Intelligence Menlo Park, CA, USA: 414-419.
Kitano, H., S. Tadokoro, et al. (1999). RoboCup Rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. IEEE SMC '99 Conference Proceedings, Vol. 6, p 739 - 743.
Klusch, M. and A. Gerber (2002). Dynamic coalition formation among rational agents. IEEE Intelligent Systems , Volume 17 Issue 3, p. 42-47.
Kraus, S., O. Shehory, et al. (2003). Coalition formation with uncertain heterogeneous information, Proceedings of the second international joint conference on Autonomous agents and multiagent systems , ACM Press New York, NY, USA: 1-8.
Kraus, S., O. Shehory, et al. (2004). The Advantages of Compromising in Coalition Formation with Incomplete Information, Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems , IEEE Computer Society Washington, DC, USA: 588-595.

Kraus, S., J. Wilkenfeld, et al. (1995). Multiagent negotiation under time constraints, Elsevier. Artificial Intelligence, vol. 75: 297-345.
Lau, N., L. P. Reis, et al. (2005). FC Portugal 2005 Rescue Team Description: Adapting Simulated Soccer Coordination Methodologies to the Search and Rescue Domain. http://paginas.fe.up.pt/~reifeup/trab_completos/FC
Lerman, K. and A. Galstyan (2003). Agent memory and adaptation in multi-agent systems, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM Press New York, NY, USA: 797-803.
Lerman, K. and O. Shehory (2000). Coalition formation for large-scale electronic markets: Proceedings of the Fourth International Conference on MultiAgent System, p. 167-174.
Lin, C., S. Hu, et al. (2007). An Anytime Coalition Restructuring Algorithm in an Open Environment, Springer. LNCS, 4681: 80.
Lin, C. F. and S. L. Hu (2007). Multi-task overlapping coalition parallel formation algorithm, Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, ACM New York, NY, USA.
Manisterski, E., E. David, et al. (2006). Forming efficient agent groups for completing complex tasks, Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems , ACM Press New York, NY, USA: 834-841.
Martin, C. (2001). Adaptive Decision Making Frameworks for Multi-agent Systems. Austin, TX, University of Texas. Ph.D.
Microsoft. (2006). "Microsoft Exchange Server: Calendar Concierge." Retrieved 5/3/2009, 2009, from http://www.microsoft.com/exchange/2007/evaluation/features/calendarconcierge.mspx.
Microsoft. (2009, 200). "Microsoft Office Project Server 2008." Retrieved 5/3/2009, 2009, from http://office.microsoft.com/en-us/projectserver/FX100739841033.aspx.
Myerson, R. B. (1991). Game theory: analysis of conflict, Harvard University Press.
Nair, R., T. Ito, et al. (2001). Task Allocation in the RoboCup Rescue Simulation Domain: A Short Note, Springer LNCS, 2377.
Nair, R., M. Tambe, et al. (2003). Role allocation and reallocation in multiagent teams: towards a practical analysis, Proceedings of the second international joint conference on Autonomous agents and multiagent systems , ACM Press New York, NY, USA: 552-559.
Nazemi, E., M. Faradad, et al. (2005). SBCe_Saviour Team Description. Tehran, Iran, Shahid Beheshti University: IEEE SMC '99 Conference Proceedings, Vol. 6.
Patel, J., W. T. L. Teacy, et al. (2005). Agent-based virtual organisations for the Grid, IOS Press. Multiagent and Grid Systems , Volume 1 Issue 4 p. 237-249.
Plunkett. (2008). "Consulting Statistics." Retrieved 5/3/2009, 2009, from http://www.plunkettresearch.com/Industries/Consulting/ConsultingStatistics/tabid/177/Default.aspx.

Rahwan, T., S. D. Ramchurn, et al. (2007). Near-optimal anytime coalition structure generation: Proceedings of the 20th international joint conference on Artifical intelligence, p. 2365-2371.
Raiffa, H. (1982). The art and science of negotiation, Belknap Press of Harvard University Press Cambridge, Mass.
Rathod, P. and M. DesJardins (2004). Stable team formation among self-interested agents. Working Notes of the AAAI-2004 Workshop on Forming and Maintaining Coalitions in Adaptive Multiagent Systems. San Jose, CA.
Saha, S., S. Sen, et al. (2003). Helping based on future expectations, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM Press New York, NY, USA: 289-296.
Sander, T., B. Peleschuk, et al. (2002). A Scalable Distributed Algorithm for Efficient Task Detection. AAMAS 2002. Bologna, Italy.
Sandholm, T., K. Larson, et al. (1999). Coalition structure generation with worst case guarantees, Elsevier. Artificial Intelligence, vol. 111: 209-238.
Sandholm, T. and V. R. T. Lesser (1997). Coalitions among computationally bounded agents, Elsevier. Artificial Intelligence 94: 99-137.
Sandholm, T., S. Sikka, et al. (1999). Algorithms for optimizing leveled commitment contracts: Proceedings of the 16th international joint conference on Artifical intelligence, p. 535-540.
Sandholm, T. and Y. Zhou (2002). Surplus equivalence of leveled commitment contracts, Elsevier. Artificial Intelligence vol.142: 239-264.
Sandholm, T. W. (1998). Contract types for satisficing task allocation: I theoretical results. In Proc. AAAI Spring Symposium: Satisficing Models 1999.
Sandholm, T. W. and V. R. Lesser (1994). Utility-based Termination of Anytime Algorithms, University of Massachusetts at Amherst, Computer Science Dept.
Sandholm, T. W. and V. R. Lesser (1995). Equilibrium Analysis of the Possibilities of Unenforced Exchange in Multiagent Systems, University of Massachusetts at Amherst, Computer Science Dept.
Sandholm, T. W. and V. R. Lesser (1996). Advantages of a leveled commitment contracting protocol: Technical Report: UM-CS-1995-072.
Sarne, D. and S. Kraus (2005). Solving the Auction-Based Task Allocation Problem in an Open Environment: Proceedings of the 20th national conference on Artificial intelligence , p.164-169.
Scerri, P., A. Farinelli, et al. (2005). Allocating tasks in extreme teams, Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems ,ACM Press New York, NY, USA: 727-734.
Scully, T., M. G. Madden, et al. (2004). Coalition calculation in a dynamic agent environment, Proceedings of the third international joint conference on Autonomous agents and multiagent systems , ACM Press New York, NY, USA.
Sen, S. and P. S. Dutta (2000). Searching for optimal coalition structures: Proceedings of the Fourth International Conference on MultiAgent Systems , p. 286-292.
Shapley, L. S. (1997). A VALUE FOR n-PERSON GAMES, Princeton University Press.

Shehory, O. and S. Kraus (1998). Methods for task allocation via agent coalition formation, Elsevier. Artificial Intelligence vol. 101: 165-200.
Shehory, O. and S. Kraus (1999). Feasible formation of coalitions among autonomous agents in nonsuperadditive environments, Computational Intelligence, vol. 15: 218-251.
Shtub, A., J. F. Bard, et al. (2005). Project management: Processes, methodologies, and economics, Pearson Prentice Hall.
Skyrms, B. and R. Pemantle (2004). , Springer, Understanding Complex Systems.
Smith, R. G. (1980). The contract net protocol. IEEE Transactions on Computers 29: 1104-1113.
Soh, L. K. and C. Tsatsoulis (2002). Satisficing coalition formation among agents, Proceedings of the first international joint conference on Autonomous agents and multiagent systems, ACM Press New York, NY, USA: 1062-1063.
Stein, S., T. R. Payne, et al. (2007). An effective strategy for the flexible provisioning of service workflows, Springer. LNCS 4504: 16.
Su, S. X., S. L. Hu, et al. (2007). Coalition structure generation with worst case guarantees based on cardinality structure, Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, ACM New York, NY, USA.
Sutton, R. S. and A. G. Barto (1998). Reinforcement Learning: An Introduction, MIT Press.
Tambe, M., D. V. Pynadath, et al. (2000). Building dynamic agent organizations in cyberspace. IEEE Internet Computing, vol 4: 65-73.
Tambe, M. and W. Zhang (1998). Towards flexible teamwork in persistent teams: Autonomous Agents and Multi-Agent Systems, vol. 3, p. 277-284.
Tosic, P. T. and G. A. Agha (2005). Maximal Clique Based Distributed Coalition Formation for Task Allocation in Large-Scale Multi-agent Systems, Proceedings of MMAS, 2004. Springer.
Wellman, M. P. (1997). Market-Aware Agents for a Multiagent World, Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Multi-Agent Rationality. Springer.
White, T. and B. Pagurek (1998). Towards Multi-Swarm Problem Solving in Networks: Proceedings of the 3rd International Conference on Multi Agent Systems 333-340.
Wilson, B. and M. DesJardins (2007). Forming Stable, Overlapping Coalitions in an Open Multi-Agent System. AAAI Fall Symposium on Regarding the 'Intelligence' in Distributed Intelligent Systems. Arlington, VA.
Wooldridge, M. and N. R. Jennings (1995). Agent Theories, Architectures, and Languages: A Survey, Springer LNCS, vol. 22.
Yokoo, M., V. Conitzer, et al. (2005). Coalitional games in open anonymous environments: Proceedings of the 20th national conference on Artificial intelligence, vol. 2.

VITA

Chris Jones was born in New Orleans, Louisiana on May 10, 1976, the son of Lyman and Jean Jones. He attended Ben Franklin High School in New Orleans, received his Bachelor's degree in Electrical Engineering from the University of Texas at Austin in 1999, and his Master's degree in Electrical Engineering from the University of Illinois at Urbana-Champaign in 2002. He has worked as a software engineer and a technical advisor for a patent firm between getting his degrees. He has been a graduate research assistant in the Laboratory for Intelligent Processes and Systems since June 2005 as a PhD candidate.

Permanent address: 116 Warren Ave N Apt 504, Seattle, WA, 98109, USA

This dissertation was typed by the author.